# MASTER ENVIRONMENTAL LIBRARY
# (MEL)

# SOFTWARE CENTER OPERATOR MANUAL
# (MEL REGIONAL SITE SOFTWARE)


**June 5, 1997**

**Defense Modeling and Simulation Office**
**Alexandria, VA**

Master Environmental Library
(MEL)

# SOFTWARE CENTER OPERATOR MANUAL
# (MEL REGIONAL SITE SOFTWARE)

**June 5, 1997**

Version 1.0

PREPARED BY:
MAR, INC.
99 PACIFIC ST., SUITE 200E
MONTEREY, CA 93940

Master Environmental Library
(MEL)

# Software Center Operator Manual
# (MEL Regional Site Software)

June 5, 1997

APPROVAL:

_____

Director,
Defense Modeling and Simulation Office

# FOREWORD

The Defense Modeling and Simulation Office (DMSO) was established to serve as the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), and to provide a full-time focal point for information concerning Department of Defense (DoD) Modeling and Simulation (M&S) activities. The DMSO promulgates M&S policy, initiatives, and guidance to promote cooperation among DoD components to maximize efficiency and effectiveness. The DMSO is a staff activity reporting to the Director, Defense Research and Engineering (DDR&E), office of the Undersecretary of Defense for Acquisition and Technology (USD(A&T)).

This document will be reviewed and updated by the DMSO as required to maintain currency. Comments and recommendations should be forwarded for review and consideration for inclusion to:

Director,
Defense Modeling and Simulation Office
1901 N. Beauregard St., Suite 504
Alexandria, VA 22311
(703) 998-0660

# LIST OF EFFECTIVE PAGES

**Page Number** **Change Number**

# RECORD OF CHANGES

| Change Number | Date of Change | Change Description | Date Entered | Entered by |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# TABLE OF CONTENTS

**NOTE: Conventions**

This manual uses the following typographical conventions:

CAPITAL LETTERS for the names of Internet protocols, acronyms, and abbreviations.

**Boldface** type for emphasis and references to other sections in this manual.

*Italicized* words have special meaning in the MEL Regional Site Software.

`Monospaced font` for keywords in computer system commands, directory path names, and file names. In proper context, the text in `[square brackets]` represents command options and text in `<angle brackets>` represents items the user should replace with applicable text.

*`Monospaced italic font`* for Internet addresses.

# SECTION 1. SCOPE

## 1.1 IDENTIFICATION

This Software Center Operator Manual applies to Version 1.1 of the Master Environmental Library (MEL) Regional Site Software (MRSS). MRSS 1.1 was released on March 24, 1997.

## 1.2 SYSTEM OVERVIEW

The MEL is an Internet-based data discovery and retrieval system that provides access to geographically-distributed oceanographic, meteorological, terrain, and near-space databases. The MEL project is sponsored by the *Defense Modeling and Simulation Office (DMSO)* under the direction and funding of *Executive Agents (EA)* for air and space, oceans, and terrain.

The MEL is based on a library paradigm in which users query a distributed *"card catalog"* located at a MEL Access Site (MEL/AS). A MEL/AS consists of an Internet HTTP Server (also known as the *Web server*) that supports Hypertext Markup Language (HTML), Java™ interfaces and supporting Common Gateway Interface (CGI) programs. The *"cards"* in the card catalog serve as the common denominator among different types of data in the library. These consist of *metadata records* that comply with the United States Federal Geographic Data Committee (FGDC) content standards for digital geospatial metadata.

MEL users search for and order available data through a World Wide Web (WWW) browser. They may choose either the HTML or Java interface to interactively create a *query*, browse the query results, and then place an order for data. The queries and orders include a region of interest, time range, category keywords, and data center elements. The query checks all metadata records at the specified data sites. Query results are displayed for the user to examine the full text of the metadata record, view any browse graphics associated with the metadata record, and link directly to the data or data site or generate an order form customized for the desired dataset. The Java query provides an interactive interface that promotes visual comparison of query results, thus guiding users through a potentially large set of criteria-fitting data to the specific datasets of interest. Users can order existing datasets, or where applicable, subscribe to regularly produced datasets for automatic distribution.

Orders for data are sent to the applicable MEL Regional Site (MEL/RS) via electronic mail (e-mail) and processed by the *MRSS*. This customizable software parses the e-mail orders, provides access control, handles scheduling of requests, extracts data from the local databases, formats, compresses, encrypts (if necessary), and delivers the data files, and finally notifies the user via e-mail of delivery.

The MRSS is written primarily in the Perl programming language. The formats in which datasets can be provided are specified in corresponding metadata records. A MEL/AS can

then generate a customized order form from these and other optional specifications, such as the facility to subset the data. The requested data is delivered in standard formats, like Gridded Binary (GRIB), Binary Universal Form for Data Representation (BUFR), and National Imagery Transmission Format (NITF), via anonymous File Transfer Protocol (FTP) or e-mail. It can optionally be stored on magnetic media and mailed to the user, or the user can have the data temporarily stored on the MEL Regional Site's anonymous FTP server for later downloading.

MEL Regional Sites maintain complete control over their data and all access to it. The MRSS supports access control, user authentication, data extraction, and data delivery. An existing data center does not need to change its data management policies or architecture when it becomes a MEL Regional Site, except that the FGDC metadata content standard must be used to describe datasets and deliver each dataset in at least one of the standard formats, as mandated by DMSO and the MEL project.

## 1.3 DOCUMENT OVERVIEW

This document identifies and describes Version 1.1 of the MRSS and ancillary support software packages. It is designed to be used for the installation, setup, customization, and administration of this software. A list of acronyms and abbreviations used in this document is provided in Appendix E.

# SECTION 2. REFERENCED DOCUMENTS

## 2.1 GOVERNMENT DOCUMENTS

### 2.1.1 STANDARDS

a. Federal Geographic Data Committee. 1994. **Content Standards for Digital Geospatial Metadata (June 8**). Federal Geographic Data Committee. Washington, DC.

b. MIL-STD-498, **Software Development and Documentation**, Department of Defense, 5 December 1994.

### 2.1.2 OTHER DOCUMENTS

a. **Digitizing the Future**, Fourth Edition, Defense Mapping Agency.

b. Executive Order 12906, Coordinating Geographic Data Acquisition and Access: The National Spatial Data Infrastructure, **Federal Register**, Volume 59, Number 71, pp. 1761-17674. 13 April 1994.

c. SPAWARSYSCOM JMCIS-DS-1.0, **Joint Maritime Command Information System Documentation Guide**, Space and Naval Warfare Systems Command, January 1995.

## 2.2 NON-GOVERNMENT DOCUMENTS

a. **Apache HTTP Server Project Home page.** Available: http://www.apache.org/info.html. 10 February 1997.

b. **free-WAIS-sf Home page.** Available: http://www.wsc.com:8080/freeWAIS-sf/aboutwais.html. 10 February 1997.

c. **MIT Distribution Site for PGP Home page.** Available: http://web.mit.edu/network/pgp.html. 12 February 1997.

d. **The Perl Language Home Page**. Available at: http://mox.perl.com/perl/index.html. 7 May 1997.

# SECTION 3. MEL REGIONAL SITE SOFTWARE SUMMARY

The MRSS serves the following two purposes:

- Electronically receive and process requests for data from users
- Deliver the data to the user while ensuring privacy and controlled access

The MRSS supports sites that have diverse resources and levels of technical expertise. The MEL Regional Sites can configure the software to suit locally available resources and data distribution policies. For example, sites may choose to manually approve requests for additional security or to automate access control for speed and efficiency.

The MRSS is a scaleable system that can support sites with various configurations. Since e-mail is used to send a user's data request to the Regional Sites, the system can work even from behind firewalls, requiring few configuration changes to the firewall setup. The user's e-mail request and, if necessary, the retrieved data files can be encrypted for privacy while being sent over the Internet. Where possible, the MRSS uses Commercial-Off-The-Shelf (COTS) software, open systems technology, and proven public domain support software.

**NOTE: What is a *request*?**

The special term *request*, as used in this document, refers to a single instance of an order for a single dataset ordered by the user. It is also assigned a unique *request_id* that is used extensively by the MRSS. Using the *request_id* the user can track the progress of the *request*, and site administrators can monitor and troubleshoot the software. The *request* is sent to the MRSS via an encrypted e-mail message that conforms to a previously defined syntax and format. The e-mail message contains information about the user, the dataset, user options, and any local processing information. For more information on the MEL e-mail order syntax, see Appendix C.

E-mail *requests* are formatted in a custom text format similar to the Standardized General Markup Language (SGML), and are processed as follows: The MEL/AS sends a formatted and encrypted user *request* via e-mail to the Regional Site. The MRSS decrypts and parses the *request*, and optionally checks for authorized access to requested datasets or information. If the *request* is determined to be legitimate, through PGP signature authentication, processing continues and the extracted data or the generated product is sent to the user via FTP or e-mail, as appropriate.

Many *requests* may be processed concurrently by the MRSS. The number of concurrent requests and the number of processes are site configurable, based on available system resources. The MRSS is composed of the following main modules:

- Scheduler (coordinate MRSS related processes)
- Request handler (decrypt and parse e-mail message)

- Access control (authorize access)
- Extractor (retrieve data from proper databases and encode into standard format)
- Postprocessor (prepare data files for delivery)
- Delivery (deliver files to user)
- Cleanup (remove all intermediate files)

Each module may be made up of one or more components consisting of separate programs and support libraries to perform necessary tasks. Each module has a distinct role in processing requests. A file called the *Job Control Block (JCB)* contains information on the *state* of every *request* currently being processed. The S*cheduler* periodically polls the JCB for changes in the *state* of the requests, and then takes appropriate action by invoking other modules.

**NOTE: What is a MRSS *state*?**

In MRSS, a *state* simply refers to what is happening to a given *request* at that time. Internally, MRSS uses a pre-defined set of constants to specify and refer to the different *states*. There are three categories of states: *IN_states, AWAITING_states, ERROR_states*. A request in an *IN_state* is being processed through one of the MRSS modules while a request in a *AWAITING_state* is waiting to be processed through the next step in the process. The *IN_states* and the corresponding *AWAITING_states* are interdependent, that is, there is a definite order and sequence for a given *request*. *ERROR_states* indicate that there is some error in the processing of the request; and further processing of the request is not possible without operator intervention.

The following sections provide overviews of the modules and their functions. More details on the functions of these modules may be found in the section on the **MRSS Architecture.**

## 3.1 SCHEDULER

This *Scheduler* provides central coordinating functions in the MRSS. It runs continuously and periodically checks the JCB, looking for a *request* that is in a *state* that needs attention for further action. The S*cheduler* first locks the JCB file and reads its entire contents. Next, it prioritizes all *requests* based on how long each has been waiting to run. It then selects the first *request* from the prioritized list, determines what further action needs to take place, logs the new *state* to the JCB, and then invokes the appropriate *module* to perform the next task. These steps are repeated until the data has been extracted, processed, delivered and the temporary files have been deleted. This process is repeated for each *request*.

## 3.2 RSS REQUEST HANDLER (RRH)

The RSS Request Handler (RRH) receives e-mail *requests* sent from a MEL/AS. The *request* is first decrypted and then parsed. Then, the RRH makes an entry in the JCB file indicating the new *state* of the *request* for the *Scheduler* to pickup on its next poll.

## 3.3 ACCESS CONTROL

The Access Control Module *(ACM)* consists of two main components: the *Client Authentication Mechanism (CAM)* and the *Data Authorization Mechanism (DAM)* . The purpose of the *ACM* is to provide facilities for *authentication* and *authorization* of the user's *request*. The CAM uses the Pretty Good™ Privacy (PGP) software and the MEL PGP Library to verify the identity and authenticity of the originator of the e-mail request.

> **NOTE:** *Authentication* involves verifying the identity and authenticity of the request, and *authorization* involves verifying the access privileges of the user.

At each MEL/RS, the DAM may be replaced by custom components.

### 3.3.1 CLIENT AUTHENTICATION MECHANISM

The authentication of the e-mail request is accomplished using the PGP software. The CAM uses a set of utility functions (called the MEL PGP Library, written in Perl) to perform the tasks. The MEL PGP Library is also used by the MEL Access Site Software (MASS) and the MEL PGP Manager.

### 3.3.2 DATA AUTHORIZATION MECHANISM (DAM)

The optional *Data Authorization Mechanism* component provides Regional Sites with complete control over MEL access to the data at its site. The DAM uses an *Access Control List (ACL)* maintained in a text file to restrict access on a per database and/or per dataset basis. The Regional Site *Administration Tool*[1] (RSS *Admin Tool*) is used to configure and maintain the *ACL*. A set of utility programs is also available to administer the *ACL*.

## 3.4 EXTRACTOR

This module links the local data extraction programs with the MRSS. It may actually consist of several data extraction programs controlled by one interface. This module must be tailored with a few lines of custom code for each extraction program integrated into the MRSS.

## 3.5 POSTPROCESSOR

After the data have been extracted, this module optionally encrypts, compresses, or otherwise prepares the data files for delivery to the user.

[1] In MRSS 1.1 the Admin Tool function for maintaining the ACL is not functional. The ACL file must be manually edited using a text editor. This procedure is discussed later in the document.

## 3.6 DELIVERY

This module delivers the extracted data to the user via FTP or other means, as specified in the User Profile. Optionally, the data may be stored on a local anonymous FTP server for later downloading by the user. If supported by the site, larger datasets may be copied to tape, CD-ROM or other media and sent by postal mail or Federal Express shipping.

## 3.7 CLEANUP

Once delivery has been made, the C*leanup* module deletes any temporary files associated with a particular *request*. It then removes all references to the *request* from the *JCB* file. The *request* response is complete once this module finishes.

# SECTION 4. MRSS ARCHITECTURE

The MRSS is composed of the seven modules described in **MEL Regional Site Software Summary,** plus the associated files and directories described in **MRSS Installation**. The following modules asynchronously process a *request* through its various stages:

- Scheduler
- Request Handler
- Access Control
- Extractor
- Postprocessor
- Delivery
- Cleanup

A description of the life cycle of a MEL *request* is provided in Appendix A. Its purpose is to assist Regional Site Administrators in understanding the overall processes of MEL.

## 4.1 MRSS DIRECTORIES AND FILES

`$RSS_DIR` This is the top level of the MRSS directory tree. It should contain subdirectories for MRSS executables, library files, log files, online documentation, and source code.

`$TMPL_DIR` This is where the temporary *template files* reside. For convenience, the contents of the e-mail message is separated into *template files* that contain all necessary information for the different MRSS modules. For each data *request* processed, the RRH creates a subdirectory using the name of the `request_id` under `$TMPL_DIR` in which to write one file for each MRSS module that contains sub-sets of information from the *request* message. These template files contain name/value pairs of information that are used as inputs to the various MRSS modules. The *RRH* creates the files when the *request* arrives, and the *scheduler* deletes them after the *request* has been successfully delivered.

`$DATA_DIR` For each incoming *request* the *RRH* creates a sub-directory using the name of the `request_id` under `$DATA_DIR` in which to temporarily store files containing the extracted data. In some cases, a link could be made from this directory to an alternate location, such as in *generic product extractors*. This method would avoid storing unnecessary duplicate files.

## 4.2 THE JOB CONTROL BLOCK (JCB)

The JCB is a text file where information about the *state* of every request being processed by MRSS is stored. The S*cheduler* refers to this file to prioritize processing actions. In addition to the *Scheduler*, this file is also accessed and modified by the other MRSS modules to update the change in *state* for each *request*. MRSS enforces a *lock mechanism* to ensure there are no concurrent accesses to the JCB. The JCB is also used by the *Scheduler* to re-start the MRSS after a shutdown or a system crash.

## 4.3 SCHEDULER

The *Scheduler* is a background process (daemon) that may be either invoked during system boot-up or run from the command line. Its purpose is to monitor changes in the *state* of a *request* and to take appropriate action. The S*cheduler* is the system coordinator. It periodically reads the JCB file, and using a modified first-in — first-out algorithm, determines which *request* to run next. A *request* in *error state* is ignored, as is a *request* whose *state* indicates that one of the other modules is already processing it. If the S*cheduler* finds a JCB entry in the AWAITING_* *state*, it calls the appropriate module and updates the *JCB* entry to IN_* *state*. The S*cheduler* logs its activities to the file `$RSS_DIR/log/scheduler.log`.

## 4.4 RSS REQUEST HANDLER (RRH)

The RRH receives *requests* forwarded from a MEL/AS. The *RRH* module performs the following tasks:

- Decrypt incoming *requests* that are PGP encrypted
- Checks for PGP signature of sending site
- Checks to see that the incoming *request* conforms to the format and syntax specified in the **MEL E-mail Order Syntax** (see Appendix C)
- Parses the *request* into an internal perl multidimensional associative array
- Creates the `template files` used by other MRSS modules
- Creates an entry in the *JCB* file for further action by the *Scheduler* and other MRSS modules

## 4.5 ACCESS CONTROL

The purpose of the *Access Control Module (ACM)* is to provide *user authentication* and *request authorization* tools and utilities for the Regional Site Administrators to implement their data access policy.

The ACM components are the *Client Authentication Mechanism (CAM)* and the *Data Authorization Mechanism (DAM)*. The ACM allows the MEL/RS to have complete control over who can access what data from the site. The access control rules are maintained in a text file called the *Access Control List (ACL)*. In MRSS 1.1, the Regional Site Administrators must use a text editor to manually configure and maintain the ACL file. In subsequent releases of the MRSS, the Regional Site *Admin Tool* will be used to maintain the ACL for local databases and datasets.

### 4.5.1 CLIENT AUTHENTICATION MECHANISM (CAM)

The *Client Authentication Mechanism (CAM)* is based on the MEL PGP Library used to authenticate incoming e-mail requests for data. The CAM performs signature checks against the PGP encrypted e-mail and the PGP public keys in the PGP *keyring*. The request is not processed further if authentication fails.

### 4.5.2 DATA AUTHORIZATION MECHANISM (DAM)

The purpose of the *Data Authorization Mechanism (DAM)* is to implement the Regional Site's data access policy. When a *request* is received, the DAM verifies a user's access to the data by consulting the ACL. The ACL file can be edited to add necessary rules and flags. The DAM interface can be replaced or augmented by local routines if desired.

A MEL/RS can configure the DAM for *manual* or *automated* access control, and also add new functionality to it. In *manual* mode, the *request* is maintained in a holding state until the Regional Site Administrator approves or denies it. In *automated* mode, the request is passed through access control tests. If access is approved, the JCB entry is updated to `AWAITING_EXTRACTION`; otherwise, the request is terminated and placed in error *status*.

## 4.6 EXTRACTOR

This module links the local data extraction programs with the MRSS. The *Extractor* may be considered a single logical module, although it may actually consist of several different data extraction programs controlled by one front-end driver.

Each of the *Extractor*'s programs must be integrated into MRSS with a few lines of Perl code for each database. These programs read from a file or from the command line, and then extract the requested data from the database. (As used here, the term *database* is a logical term defining a collection of datasets, regardless of how they are stored or retrieved.) The *Extractor* module contains one section common to all sites and one or more sections customized for each of its local databases.

The *Extractor* reads the contents of the extraction template into a perl associative array (a hash table of key/value pairs). From this template it determines what dataset the user has requested. If necessary, an input file is created for the requested dataset. A small section of custom code performs a logical check against valid dataset names and invokes the corresponding extraction program with an input file to retrieve the data. The input file(s) for the database extractors need to be generated using Perl code from the *Extractor* module. In many cases, it is convenient to integrate the data format encoders into the extraction programs.

The retrieved records are stored in files awaiting transfer to the user, and their *states* are changed in the *JCB* to `AWAITING_POSTPROC`. If the data was not available or could not be extracted, an e-mail message is sent to the user, the *request*'s status is changed to `ERROR_DEBUG,` and an e-mail is sent to the MEL Administrator with information regarding why the job failed.

## 4.7 POSTPROCESSOR

After the data has been extracted, this module encrypts, compresses, or performs other data preparation functions that the user has specified. The *Postprocessor* then reads its template file to obtain specifications for processing the extracted data files. These specifications are:

`TAR [YES | NO]` – Option to combine or not combine the files into one archive.

`COMPRESS`[2] `[NONE | UNIXCOMPRESS | GZIP | PKZIP]` – If specified, TAR is performed, followed by COMPRESS, using the specified method. Many data formats are already compressed and further compression is not possible.

`PGP_ENCRYPT [YES | NO]` – As an option, files may be PGP encrypted using the user's certified public key so that only the holder of the user's secret PGP password can decrypt the data.

After the *request* has been processed, the *Postprocessor* changes the *request status* in the *JCB* to `AWAITING_DELIVERY`.

## 4.8 DELIVERY

After post-processing, the *Delivery* module reads the delivery template to obtain the delivery address and method. It then delivers the extracted data to the specified address. The delivery options are:

FTP: Delivery to a user site via FTP.

FTP_PKUP: Delivery to a Regional Site's anonymous FTP server, where the user can download the requested data at a later time.

[2] This feature is not supported by the MASS 1.1.

The user is notified of delivery by e-mail. If delivery to a user's anonymous FTP site fails, the data is placed on the Regional Site's anonymous FTP server and the user is notified of the file name and location. Finally, the *JCB* status is changed to `AWAITING_CLEANUP`.

## 4.9 CLEANUP

The *Cleanup* module deletes the temporary *control files* associated with a particular *request*, and then removes the *request* from the *JCB*. An entry about the request is made in MRSS log files.

# SECTION 5. MRSS INSTALLATION

The installation and configuration of the MRSS software is a complex process[3]. A complete listing of the files needed for the MRSS installation is provided in Appendix B. If problems are encountered, please consult **Troubleshooting** before contacting the MEL support staff.

## 5.1. REGISTRATION

All MEL Regional Sites must be approved by the Oceans Executive Agent. Data sites desiring to become a MEL Regional Site should contact the Oceans EA at the following Web site:

*http://www.dmso.mil/ocean_ea/OceanHome.html*

## 5.2 SITE PREREQUISITES AND SUPPORT

Although MEL Regional Sites are essentially autonomous, they must meet the following minimum requirements to provide platform-independent access and ensure users will be satisfied with the reliability and efficiency of the MEL data discovery and delivery system.

### 5.2.1 SITE ADMINISTRATOR

Prospective MEL Regional Sites must identify a Regional Site Administrator, whose administrative responsibilities will include:

- Administration and maintenance of the MEL Regional Site hardware and software
- Creation and management of access groups and the access control file (if necessary)
- Management of site activity to ensure reliability and availability of data. This responsibility includes using the *MEL Administration Tool* to monitor *requests*, authenticate PGP keys, manual approval of requests (if necessary), maintain a Web server, and monitor MRSS system logs.

MEL Regional Site Administrators should be familiar with the fundamentals of UNIX and how the Internet works.

### 5.2.2 SITE MEL DATABASE SPECIALIST

The Database Specialist is responsibile for the integration of the database extractors into MRSS and the development of any necessary encoders and decoders to convert

[3] It is not unusual to spend 40 hours or more installing and configuring the MRSS.

data from native formats to MEL approved standard formats. Often, the same person may function as the MEL Regional Site Administrator as well as the MEL Regional Site Database Specialist. This person must be comfortable working with MEL datasets and managing them. A working knowledge of Perl is necessary to integrate the local data extractors to the extractor module in the MRSS.

### 5.2.3 SYSTEM PREREQUISITES

Prospective MEL Regional Sites should have the following hardware, software, and network components installed and tested *before* attempting to install the MRSS. The technical personnel should understand how these components work, especially the PGP and HTTP server software.

**Hardware**

- Recommended Random Access Memory (RAM) - 128 MB
- Recommended available Hard Disk Capacity – 3 hard disks, each of 9 GB storage capacity. All three disks should be of the same make and model. One of these disks should be kept off-line as a spare.
- Recommended Processing Power - Computer class equivalent to or better than Sun SPARC 20.

> **NOTE: Other Sites.**
>
> Hardware successfully used by other sites include SGI/Indigo2 R4400, TAC-4, DEC Alpha 1000 server, SGI Onyx, Sun SPARC 20, and Sun SPARC Center 1000 server.

**Network**

- Sufficient bandwidth on Local Area Network (LAN) and Internet connections to handle forecast peak loads with acceptable response times for FTP deliveries and WAIS/Isite server queries
- Highly reliable and available network
- Reliable e-mail

**NOTE: Firewalls and SMTP**

If the site is behind a firewall, and the firewall/gateway refuses incoming SMTP connections, the firewall administrator should e-mail the MEL MRSS Support Team at `mel_support@nrlmry.navy.mil` with a description of the problem. There may be other means of forwarding mail to the MRSS server from inside or outside the firewall.

**Software**

The following support software tools are required to install and customize the MRSS. Some of these may already be available on the system. *At present, MRSS is supported only on UNIX platforms.*

- GNU Zip utility (gzip). This utility is needed to uncompress some of the software files. It is available from:

  `ftp://prep.ai.mit.edu/pub/gnu/gzip-1.2.4.tar`

- HTTP server (Web server). *Required by the MRSS Admin Tool and other utilities*. There are a number of excellent choices for UNIX:

  - The National Center for Supercomputer Applications (NCSA) server from:

    `http://hoohoo.ncsa.uiuc.edu/docs/setup/OneStep.html`

  - The Apache server (recommended) from:

    `http://www.apache.org/`

  - The European Laboratory for Particle Physics (CERN) server (also known as the W3C server) from:

    `http://www.w3.org/pub/WWW/Daemon/`

  - Netscape Commerce Server™ from:

    `http://www.netscape.com/`

**NOTE: Security and HTTP Server Configuration**

Site personnel should become familiar with the configuration procedures for these servers since they can be a security risk if improperly configured. Consult the documentation that comes with the server software listed above. An HTTP server alias should be established to ensure users can reliably access the data over a period of years; in this way, when the server is changed, the alias can be moved without disrupting established user links to the data. Also, the effective User Identification (UID) of the server processes should be the same as that of the MRSS or steps should be taken to ensure the server will be able to access and modify the MRSS resources such as the *JCB* and the log files.

- WAIS server and indexing tools. This is used for indexing and serving FGDC-style metadata to the Internet. The recommended version is `freeWAIS-sf-2.0.65`.

  The recommended procedure for first-time installers is to have the MEL Access Site Administrator create a WAIS database on the MEL Access Site computers and remotely host the Regional Site's metadata records. Although the metadata records are hosted on the MEL Access Site WAIS server, the Regional Site Administrator is still responsible for validating and maintaining their metadata records.

**Note: Planned Upgrades to Isite**

MEL plans to migrate to a new generation of software, called *Isite*, to replace the current *freeWAIS*. The *Isite* software is the successor to WAIS. *Isite* is expected to be more robust and flexible in supporting geospatial queries. The target date for the switch over is Fall 1997. Further details will be announced in the proper MEL e-mail lists.

- Encoder/decoder libraries as needed; these are available on the MEL Web site via:

*`http://www-mel.nrlmry.navy.mil/software.html#encoders_decoders`*

- The MRSS is written in the Perl language, and requires Perl 5.003 or higher. Although the latest release of Perl may work, Perl 5.003 is recommended for use with MRSS 1.1.

Perl 5.003 is available at:

*http://www.perl.com/CPAN/src/5.0/perl5.003.tar.gz*

The latest version of Perl is available at:

*http://www.perl.com/CPAN/src/5.0/latest.tar.gz*

**NOTE: Perl Installation Shortcuts**

To save time when running the Perl installation configuration program one may use the default options, then follow the instructions below to compile, test, and install the software:

```
$ make all
$ make test
```

If no errors are reported, type

```
$ make install
```

to complete the installation.

MRSS expects to find the `perl` executable binary in the `/usr/local/bin` directory. If the `perl` executable is in some other location, make a link to it from `/usr/local/bin`.

```
$ ln -s <path_to_perl_executable>/usr/local/bin/perl
```

- CGI.pm is a perl module used by the *Admin Tool*. It is available at:

  *http://www.perl.com/perl/info/cool_modules.html*

**NOTE**: CGI.pm has to be reinstalled if Perl is reinstalled or upgraded.

- The MRSS uses the PGP 2.6.2 or higher encryption/decryption software to provide security and privacy by encrypting and/or decrypting e-mail *requests* and delivered datasets. MEL Regional Site Administrators should have a working knowledge of PGP and a fundamental understanding of public key encryption. Regional Site Administrators should also be aware of explicit software export control restrictions in the PGP software. PGP is available at:

  *http://web.mit.edu/network/pgp.html*

  Additional information on this subject may be found in **PGP Installation and Setup**.

### 5.2.4 DATA FORMAT REQUIREMENTS

There are two sets of data format requirements, one for the metadata that describe the individual datasets and the other for the delivery of the data itself to the user.

a. The FGDC Content Standards for Digital Geospatial Metadata **must** be used to describe datasets that are made available through MEL. The FGDC metadata standard is available at:

*ftp://www.fgdc.gov/pub/metadata/meta6894.txt*

The metadata standard provides a common set of terminology and definitions for documentation of geospatial data. It establishes the names and hierarchy of data elements to be used for these purposes, the definitions of these data elements and groups, and a legal domain of values that are to be provided for the data elements. The standard also establishes rules concerning which terms are mandatory or optional (provided at the discretion of the data provider). The FGDC Geospatial Data Clearinghouse Activity Web site provides extensive information, resources, and support for implementing the metadata standard. It is available at:

*http://www.fgdc.gov/clearinghouse/index.html*

b. Where applicable, data must be delivered to users in one or more of the DMSO approved formats for MEL. The approved formats for MEL delivery are GRIB, BUFR, and NITF. If the data cannot reasonably be delivered in any of the standard formats, arrangements may be made to support alternate formats, provided the alternate format is first approved by the DMSO and MEL Project Manager. For further information and guidance on this subject, contact MEL MRSS Support at:

`mel_support@nrlmry.navy.mil`

Further information on data content and format standards can be found in the **MRSS Standards and Data Formats**.

## 5.3 INSTALLING THE SOFTWARE

MRSS uses many directories to store and manage the software and temporary files that are created while processing each request. The three categories of these directories are: `$RSS_DIR, $TMPL_DIR, and $DATA_DIR`.

A MEL user *account* must first be created for the MRSS installation. In the examples that follow, `mel` is used as the MEL user account and the MEL *home* directory is represented as `~mel`. Sites must also create a user *group* for MRSS related files and use this as the default *group* for `mel`.

In the examples that follow, the MRSS will be installed in this user account. The MRSS is packaged as a directory tree whose root has a name indicating the version of software, `rss<version>.` The version number 1.1 will be used in these examples. This directory tree is typically installed directly under the `mel` home directory (`~mel/rss1.1`). The top level of the MRSS software directory will be referred to as `$RSS_DIR` in the following examples.

The following steps must be performed to install, configure, and test the MRSS:

Step 1. Download and install the MRSS

Step 2. Set up the MRSS *Administration Tool*

Step 3. Configure MRSS for your site

Step 4. Test the MRSS installation

Step 5. Start the *Scheduler*

Step 6. Contact the MEL Access Site Administrator

## STEP 1: DOWNLOAD AND INSTALL THE MRSS.

a. Download the MRSS distribution file from:

*http://www-mel.nrlmry.navy.mil/users-bin/order_rss*

The file is called `rss*.tar.Z`, where * is the version number of the release (Version 1.1 is used in the following examples).

b. Uncompress and untar the `rss1.1.tar.Z` file:

```
$ uncompress rss1.1.tar.Z
$ tar -xvf rss1.1.tar
```

`~mel` now contains a `~mel/rss1.1` directory and a number of subdirectories. Hereafter, the instructions will refer to this `~mel/rss1.1` directory as the `$RSS_DIR` directory. The version number 1.1 is used in these examples.

c. Change to the `$RSS_DIR` directory:

```
$ cd <$RSS_DIR>
```

d. Run the installation program:

```
$ install_rss
```

This completes the basic installation of the software. The remaining steps describe how to set up and customize the software for each site.

## STEP 2: SET UP THE MRSS ADMINISTRATION TOOL.

The Regional Site *Administration Tool* (*Admin Tool*) is a collection of programs, mostly perl scripts, located in the `$RSS_DIR/cgi-bin/` directory. This suite of programs allows Regional Site Administrators to interactively monitor site activity, subscriptions, and user data access authorization. The *Admin Tool* requires an installed and operating HTTP server for its operation.

The *Admin Tool* is accessed from a Web browser such as Netscape Navigator™ or Microsoft Internet Explorer™ and uses an HTTP CGI interface. The HTTP server at each Regional Site must have access to the scripts located in the `$RSS_DIR/cgi-bin` directory. That means the `$RSS_DIR/cgi-bin` directory needs to be 'visible' to the HTTP server software so it can read and execute these scripts. Similarly, the *JCB* should have file permissions set to allow the *Admin Tool* to read, edit, and modify the *JCB*, which means the *JCB* should have read and modify file permissions set for the HTTP server. The HTTP server must be run with the same ownership and group privileges as the MRSS (that is, the server process `httpd` and all its `children` should have an effective `UID` of the MEL user account – `mel`). Moreover, it must be specified in the HTTP server configuration files that scripts can be executed via the CGI interface. On the NCSA and Apache servers, this can be done by editing the configuration files `access.conf` and `srm.conf`. Similarly, the online documents are stored in the `$RSS_DIR/doc/html` directory. This directory also must have an `alias` in the HTTP server configuration files for the server to allow access to these files. Make sure to expand `$RSS_DIR` to its full path in the instructions below and elsewhere. The following example shows how to create these aliases for the Apache web server. For other servers, consult the server configuration documents and follow similar steps to set up the necessary aliases.

a. Edit `$RSS_DIR/lib/header.pl` to make the minimal changes that would make the *Admin Tool* functional. Later, the *Admin Tool* may be used to edit the rest of the site configurable options contained in `header.pl`.

   Change `$CGI_URL` to an HTTP server alias for `$RSS_DIR/cgi-bin`

   Change `$DOC_URL` to an HTTP server alias for `$RSS_DIR/doc/html`

b. Move to the directory where the HTTP server configuration files are located:

```
$ cd <HTTP_config_directory>
```

c. Edit `httpd.conf` to run the HTTP server as `user mel` and `group mel`:

```
User mel
Group mel
```

d. Set up an HTTP server `alias` to point to the MRSS CGI and HTML directories.

Edit `srm.conf` to add:

```
ScriptAlias /<your_CGI_alias>/ $RSS_DIR/cgi-bin/
Alias /<your_document_alias>/ $RSS_DIR/doc/html/
```

e. Edit `access.conf` to allow execution of scripts in `$RSS_DIR/cgi-bin` and to allow access to the online documents in `$RSS_DIR/doc/html`.

```
<Directory $RSS_DIR/cgi-bin>
    AllowOverride None
    Options ExecCGI
</Directory>
<Directory $RSS_DIR/doc/html>
    AllowOverride None
</Directory>
```

> **NOTE**: Restart the HTTP server for the changes to take effect.

f. Edit the file `$RSS_DIR/lib/mel_admin_users.list` to add yourself and other authorized users. Access to the tool is restricted to those on the list. Insert the Internet Protocol (IP) Address and password for each machine from which access is required. These passwords should *NOT* be the same as the login password, and should be used only for accessing the *MRSS Admin Tool*.

g. Start the MRSS *Admin Tool* from your Web browser by specifying the Universal Resource Locater (URL):

```
<$CGI_URL>/AdminTool
```

where `$CGI_URL` is `http://<your-server>/<your-cgi-alias>`

## STEP 3: CONFIGURE MRSS FOR YOUR SITE.

The MRSS may be configured to suit your requirements by editing the various site configuration variables in the `header.pl` file. The *Admin Tool* is recommended for this purpose. A mechanism must also be set up to redirect the incoming e-mail requests to the MRSS as explained below:

a. Update the set of system *variables, pathnames, and preferences* for that site. Start the *Admin Tool* and either select `Reconfigure RSS`, or edit the `$RSS_DIR/lib/header.pl` file with a text editor.

**NOTE: Perl Programming Tip**

If the *Admin Tool* is not used to edit `header.pl`, verify no syntax errors occurred during editing. When finished making the changes, run Perl on `header.pl` to check for syntax errors:

```
$ perl  -c  header.pl
```

b. Redirect incoming e-mail to the *RRH*. There are two ways of doing this: (1) create a `~/.forward` file, or (2) edit the `/etc/aliases` file used by the mail programs.

In the `~mel` directory, create a `.forward` file and add the following line:

```
| <$RSS_DIR>/bin/mel_rrh
```

OR, add the following to the file `/etc/aliases`:

```
<rss_e-mail_id>: "| $RSS_DIR/bin/mel_rrh"
```

where `rss_e-mail_id` is the e-mail address (without the *domain name*) to which the *requests* are sent. For example, one may create an alias called ***`mel-request`*** and all e-mail that is sent to ***`mel-request@your_site.your_domain`*** would then be redirected into ***`mel_rrh`***.

**NOTE: Changing the `Aliases` file**

If any changes have been made to the `/etc/aliases` file, run the program `newaliases` for the changes to take effect.

## STEP 4: TEST THE MRSS INSTALLATION.

To verify the software has been properly installed, do the following:

a. Edit the test e-mail message located in the file `$RSS_DIR/bin/mail/test.mail` to add the Site Administrator's e-mail address to the file (perhaps <u>your</u> e-mail address, if you are the Site Administrator. DO NOT use the ***`mel@hostname`*** e-mail address because it would create a feedback loop and crash the system). Also, provide a proper URL for delivering the test data files.

b. Now redirect the test e-mail to the *RRH*:

```
$ cd <$RSS_DIR/bin>
$ ./mel_rrh < $RSS_DIR/bin/mail/test.mail
```

This simulates an incoming e-mail, sending the contents of the file `test.mail` via e-mail to the *RRH*.

c. Check e-mail messages and look for a message from MRSS which contains a reference number (also referred to as request_id). The request_id has the format:

`job-id: <$SITE>-x123456,` where `$SITE` is defined in `header.pl` file.

Make a note of it. The x before the number string indicates that the *request* did *not* come from a MEL/AS. If you receive a error message, refer to the **Troubleshooting**.

d. Look in the `$RSS_DIR/tmpl/` directory for a new directory named with the above *request id* number. There should be four files: `acl.tmpl, delivery.tmpl, extract.tmpl, and postproc.tmpl`. If the files are not present, a proper installation was not completed and troubleshooting is required.

**NOTE:** ***Admin Tool*** **Tip.** Make a browser bookmark for this URL to will provide quick access to the essential *Administration Tool*.

e. If an entry for your data *request* is not there, check the `.forward` file in the *home* directory to see if it contains the complete and correct path to the *RRH*: `|$RSS_DIR/bin/mel_rrh`.

f. If set up was complete, the *RRH* should have added this test request to the *JCB* to be processed. Select '*View JCB*' in the *Admin Tool* to view the contents of the *JCB*. This request will not be processed further until the *scheduler* is started.

## STEP 5: START THE *SCHEDULER*.

The MRSS *Scheduler* coordinates the activities of the MRSS modules. The following instructions set the *Scheduler* to operate in the foreground.

a. Start the *Sheduler* in the foreground:

```
$ cd $RSS_DIR/bin
$ ./mel_scheduler  -start
```

b. As the MRSS begins to process your simulated test e-mail request, you will see text output which includes the word *starting...,* followed by scrolling console messages. Meanwhile, in your Web browser, the MEL *Admin Tool* will display and update the various *states* of the *request* as it is being processed by the various modules. The display stops changing after the *request* has been delivered.

c. Check e-mail messages. There should be a message confirming delivery, giving the request identification number, and specifying how the data was delivered.

d. Check to see if the data has been delivered to the FTP directory specified in the e-mail. The e-mail message includes the FTP URL to which the test data has been delivered.

e. Go to the FTP directory specified in the e-mail and untar it. Locate a file named `data.out,` this is the test data file for your *request.*

```
$ tar -xvf  <filename.tar>
$ more data.out
```

f. Stop the *Scheduler* using `<CTRL-C>.`

## STEP 6: CONTACT THE MEL ACCESS SITE ADMINISTRATOR.

To complete the installation, send the following information to your MEL Access Site Administrator at `mel_access_admin@nrlmry.navy.mil`.

a. The URL of the `meljobs.cgi` script. Include the port number if your HTTP server's default port is other than 80. Example:

*http://<RSS-web-server:port>/<your_CGI_alias>/meljobs.cgi*

b. The e-mail address to which MEL Access Sites should send requests. Example:

*mel@yoursite.domain*

c. The name of your site as you want it to appear on the *request* order page. *The shorter the better.*

d. Your MEL server's PGP public key (See PGP Installation and Setup).

e. Your (Regional Site Administrator's) PGP public key (See PGP Installation and Setup).

The MRSS has been successfully installed. The next step is to customize this installation for your site.

# SECTION 6. PGP INSTALLATION AND SET UP

The use of PGP is mandatory for MRSS version 1.1. MRSS will not work correctly without properly installed and configured PGP software. If site administrative personnel are not yet familiar with PGP fundamentals, they should read the PGP documentation available from:

*http://web.mit.edu/network/pgp.html*

Before configuring PGP, make sure you have:

- Downloaded and installed PGP according to the documentation instructions accompanying the software. PGP is available from:

  *http://web.mit.edu/network/pgp.html*

- Read the PGP documentation and gain an understanding of *public* and *private keys, PGP signatures,* and *keyrings* and know how to create, add, and delete them. Additional help on using PGP is available at:

  *http://www.efh.org/pgp/pgpwork.html*

Set up PGP for MRSS by completing the following six steps:

Step 1. Create a PGP directory for each PGP user

Step 2. Create public and secret key pairs

Step 3. Extract your public key(s) for distribution and MEL use.

Step 4. Add the MEL Access Site's PGP public key to your keyring

Step 5. Configure the location of PGP

Step 6. Verify the Key

## STEP 1: CREATE A PGP DIRECTORY FOR EACH PGP USER

A *PGP* directory containing the *public* and *secret keyrings* must be created in the *home directory* of each PGP user on the system. The MRSS requires a PGP setup for the MEL user account. In addition, the site administrator will also need to repeat this for the administration account. This directory is usually named `~mel/.pgp` (note the dot preceding the pgp). Only the owner of the `~/.pgp` directory should be able to access its contents – specify read/write/execute permissions.

```
$ chmod 700 ~/.pgp
```

In addition to the *keyrings*, the directory must also contain a PGP configuration file, usually called `config.txt`. A sample `config.txt` file is available with the PGP software; otherwise refer to the PGP documentation or 'man' pages for format details. To set the locations of *pubring*, *secring*, and *randseed*, copy and edit this file as follows:

a. Copy the `config.txt` file to `~/.pgp` and edit as explained in the steps below.

b. Comment out `Armor = on`, otherwise large data files will be split into sets of smaller files before encryption.

c. Set `MyName = <your_MEL_user_name>`, which is the name it uses to find your regional server's keys on your *keyrings*.

d. Add the following to `config.txt`:

```
verbose = 0
pubring = <MEL_home_directory>/.pgp/pubring.pgp
secring = <MEL_home_directory>/.pgp/secring.pgp
randseed = <MEL_home_directory>/.pgp/randseed.bin
```

## STEP 2: CREATE PUBLIC AND SECRET KEY PAIRS

Before creating a PGP *public* and *secret key* pair, you need a unique PGP *key_id* (call it *pgp_user_id*).

For example, if your *Internet domain name* is `nrlmry.navy.mil` and the MEL account *user name* for your MRSS installation is `mel`, then your PGP *pgp_user_id* would be:

```
mel@nrlmry.navy.mil
```

Now, create *public* and *private key* pairs for your MRSS server:

- Invoke PGP to create keys and follow the instructions from PGP

  ```
  $ pgp -kg
  ```

- Select option 3 and 1024 bits - Military grade - slow, but highest security
- Enter the user *pgp_user_id* for your *public key*
- Enter your PGP pass phrase. **Write it down, keep it safe, and do not lose it.**
- Enter random keystrokes until you hear the beep
- Your server's *public* and *secret key pair* will automatically be installed on your *public* and *secret keyrings* respectively

View your new key:

```
$ pgp -kv
```

### STEP 3: EXTRACT YOUR PUBLIC KEY(S) FOR DISTRIBUTION AND MEL USE.

a. Extract your key and place it in a common directory for the MEL RRH to use when sending acknowledgments using:

```
$ pgp -kxa <your_pgp_user_id> $RSS/runtime/pgp.key
```

Where <your_pgp_user_id> is the PGP User ID of the MRSS at your site.

> **NOTE**: The `-kxa` option will automatically append the `.asc` extension to the file.

b. E-mail the key to the *MEL Access Site Administrator* *(`mailto:mel_access_admin@nrlmry.navy.mil`)*. Be sure to include your area code and telephone number so the key can be verified over the phone.

c. Repeat Step 2. and Step 3. to create keys for your Regional Site administrator's user account. This is not required for installation of the RSS, but will allow your Regional Site Administrator to communicate with the MEL System Administrator in a secure manner if needed.

### STEP 4: ADD THE MEL ACCESS SITE'S PGP PUBLIC KEY TO YOUR KEYRING

a. Download and save the MEL Access Site *public key* from:

*`http://www-mel.nrlmry.navy.mil/pgp_keys`*

b. Add this key to your *keyring*:

```
$ pgp -ka <filename>
```

c. You will then be asked, "`Do you want to certify any of these keys yourself (y/N)?`" - enter 'y'

d. The fingerprint for the MEL access key will be shown on the screen. Contact the MEL Access Site Administrator for assistance in certifying the fingerprint of the MEL access key. If the fingerprints match, then answer the question "`Do you want to certify this key yourself (y/N)?`" with a '`y`'.

e. If you are certain that the key belongs to the specified user, then answer with a `'y'(Yes)` while certifying the public keys of trusted parties. Using the telephone to verify the fingerprint is a safer practice than using e-mail.

f. Enter the pass phrase for your `'mel'` user name

g. Indicate your level of trust by entering a `'4'` to grant access

## STEP 5: CONFIGURE THE LOCATION OF PGP

Edit `$RSS_DIR/lib/pgppass.pl,` setting the value of `$pgppass` to the Regional Site Software's *PGP pass phrase*.

> **CAUTION:** The presence of the pass phrase in the software code represents a potential security vulnerability, **ensure only the MRSS has permission to read these files**.

Edit $RSS_DIR/lib/pgppass.pl and header.pl to set $PGPPASSPL, $PGPPATH, $ENV{'PGPPATH'} and $keyring as required.

Run `perl -c` on each of these modules to catch any syntax errors after you edit them.

```
$ perl -c $RSS_DIR/lib/header.pl

$ perl -c $RSS_DIR/lib/pgppass.pl
```

> **CAUTION:**
>
> In its present configuration, the MRSS software assumes that any key added to the public keyring can be trusted. If an order requests encryption and the customer's PGP public key is on your public keyring, the MRSS will automatically encrypt data and send it to the person who requested the data. **Verify** all keys immediately after placing them on your keyring. **This is important because an impostor could have substituted his key for your customer's**. In such cases, it would appear that you were encrypting data to a valid customer, but you would actually be sending it to an impostor.

## STEP 6: VERIFY THE KEY

To verify a *public key* over the phone:

a. Verify fingerprint:

Call the PGP user, whose key(s) you are adding to the appropriate keyrings. Both the user and the Administrator needs to extract the fingerprint by typing:

```
$ pgp -kvc <pgp_user_id>
```

This displays the key's 16-byte digest of the *public key*.

Ask the customer to read the fingerprint. If they match, you can sign the key.

b. Sign the key:

```
$ pgp -ks <customer_pgp_user_id> -u <RSS_pgp_user_id>
```

Signing a key indicates that, based on your first-hand knowledge, you are sufficiently convinced to certify that the *public key* belongs to the identified user.

c. Edit the trust parameters:

```
$ pgp -ke <customer_pgp_user_id>
```

Here you render your opinion as to whether you trust this person to introduce and certify another person's *public keys* to you.

In the future, you may want to add additional keys to your keyring. Add and verify each new key as follows:

```
$ pgp -ka <pgp user_id>
```

# SECTION 7. CUSTOMIZING THE MRSS

> **NOTE**: Before taking these steps, you might want to review the section **MRSS Architecture** for an in-depth look at the MRSS software and how it works.

Before customizing the MRSS, both the MRSS and PGP must have been properly installed and tested. The following steps will customize the MRSS for a site:

Step 1. Edit the `header.pl` file

Step 2. Start the *`scheduler`*

Step 3. Link local database extractors to the MRSS

## STEP 1: EDIT THE HEADER.PL FILE

Edit the file `header.pl` to set values for the configuration variables. To accomplish this, use a favorite UNIX editor or the *Admin Tool*. The most current definitions are also available in the *Admin Tool*. Use the following descriptions as a guide:

**REGIONAL SITE IDENTIFIERS:**

| | |
|---|---|
| `$SITE` | The unique four letter site identifier assigned by the MEL Access Site administrator. It is prefixed to `REQUEST-ID` on data *requests* sent to your site. |

**REGIONAL SITE SYSTEM DIRECTORIES:**

| | |
|---|---|
| `$MEL_DIR` | The Regional Site software directory (`$RSS_DIR`). Type the full path: Example: `/users/mel/rss1.1` |
| `$LIB_DIR` | Location of MRSS libraries and modules; usually `$RSS_DIR/lib` |
| `$DATA_DIR` | Directory where extracted data is to be temporarily stored before delivery. |
| `$TMPL_DIR` | Directory where template files would be located. |
| `$JCB` | Location of the Job Control Block file. Usually, `$RSS_DIR/runtime/JCB` |

| | |
|---|---|
| `$FTP_DIR` | Directory where data is stored at your site for customer pickup via FTP (`FTP_PKUP`). |
| `$FTP_URL` | Address of the `FTP URL` for FTP_PKUP. Set this to a public, outgoing FTP directory at your site. Example: *`ftp://machine.domain/send/pkup-directory`* |
| `$USMAIL_DIR` | Directory where data is stored awaiting U.S. Mail delivery. |
| `$FILE_BASE` | Path to where you store your *generic file extractor (GFE)* data files and products. |

**PATHS TO EXECUTABLES:**

| | |
|---|---|
| `$TAR` | Full path to the `tar` program |
| `$COMPRESS` | Full path to the `compress` program |
| `$GZIP` | Full path to the `gzip` program |
| `$PKZIP` | Full path to the `pkzip` program |
| `$MAILPROG` | Full path to the mail agent (`sendmail`) program |
| `$PGPPATH` | Full path to the PGP program (Example: `/usr/local/bin/pgp`) |
| `$PGPPASSPL` | Location of MRSS configuration file for PGP. Usually, `$RSS_DIR/lib/pgppass.pl` |

**SET PREFERENCES:**

| | |
|---|---|
| `$MEL_ADMIN` | The e-mail address of the Regional Site Administrator to whom MRSS sends error messages. Do **not** set this to user "`mel`" which could result in a feedback loop and crash the system. |

**CAUTION:** The @ ***must*** be preceded by a backslash (\).

Example: *`administrator\@hostname.domain`*

It will be convenient to create an *e-mail alias* (for example, `mel-admin`) to whom any relevant MRSS operational messages could be sent.

| | |
|---|---|
| $JOB_WAIT | Number of seconds the *scheduler* waits before checking the *JCB* for new *requests*. |
| $DELAY_SECONDS | The number of seconds to allow a *request* to sit idle before notifying the Site administrator. |
| $MAX_CHILDREN | Maximum number of child processes (data *requests*) the *scheduler* can have active at one time. |
| $MAX_EXTRACTIONS | Maximum number of *extractor* processes that the *scheduler* can run at one time. |
| $MAX_DELIVERIES | Maximum number of delivery processes that the *scheduler* can run at one time. |
| $DEBUG | Set to 1 for verbose messages; set to 0 for fewer messages. |
| $PGP_SIG_CHECK | Set to 1 by default, meaning PGP is on. Set it to 0 to turn it off; but PGP is mandatory as of RSS1.1. So the value must be 1 for normal operations. |
| $CHECK_ACL | May be either manual or auto. |
| $ACCESS_FILE | Location of the access control file. Usually, $RSS_DIR/runtime/DAM.list |
| $SALT | A two character string used by password routines. |

**ADMINISTRATION AND DOCUMENT URL:**

| | |
|---|---|
| $CGI_URL | URL address of the cgi-bin MEL Site administrative executables: Example: *http://machine.domain/<your_CGI_alias>* |
| $DOC_URL | URL address of the MEL Regional Site Administrator's Guide files: Example: *http://machine.domain/<your_document_alias>* |

## STEP 2: START THE *SCHEDULER*

### RUNNING THE *SCHEDULER*

To run the scheduler automatically at system start-up, add a line such as this to your start-up routine:

```
$RSS_DIR/bin/mel_scheduler -start
```

The name and location of the start-up file will vary depending on the version of UNIX being used, but it should not be started as `root`. Since the s*cheduler* executes several other modules during normal operation these other modules will also run with the same UID and permissions as that of the *scheduler*. Running the *Scheduler* as `root` would be an unnecessary security risk.

The *Scheduler* may be run either in the background or foreground. By running it in the foreground, you can monitor the standard output and standard error of the *Scheduler* itself as well as modules it executes. The advantage of running it in the background is that it does not shut off when you log off your system. It normally should be run in the background, except when you are troubleshooting a problem which otherwise could not be solved.

To run the *Scheduler* in the foreground, type:

```
$ mel_scheduler -start
```

To run it in the background, type:

```
$ mel_scheduler -start &
```

**SCHEDULER USAGE**

```
mel_scheduler [-stop |-start |-reconfig |-boot |-help |-verbose][ -srb ]
```

The `mel_scheduler` recognizes the following options:

| | |
|---|---|
| `-stop` | Shut down the *scheduler*. Delete `mel_scheduler.pid` and `JCB.lock.` |
| `-s` | Same as `-stop` |
| `-start` | Use this option to start the *scheduler* from the command line. This option allows the viewing of all the standard output of processes forked by the *scheduler*. |
| `-run` | Same as `-start` |
| `-reconfig` | Tells the running *scheduler* to reread `header.pl`. Use this option after changing configuration parameters in `header.pl.` |
| `-r` | Same as `-reconfig` |
| `-boot` | Use this option if you are starting `mel_scheduler` from `inittab, init.rc*` or other start-up files. It deletes old `scheduler.pid` files and *JCB* lockfiles. Use this option *only* when starting from start-up |

files; otherwise, deleting `scheduler.pid` or *`JCB`*`.lockfiles` could result in corrupted data.

| | |
|---|---|
| `-b` | Same as `-boot` |
| `-help` | Screen describing options |
| `-h` | Same as `-help` |
| `-verbose` | Enable verbose debugging messages |
| `-v` | Same as `-verbose` |

## STEP 3: LINK LOCAL DATABASE *EXTRACTORS* TO THE MRSS

To extract data from the site's database and deliver it to customers via the MRSS, you must link your local site's *data extractors* to the MRSS. A *data extractor* is a program that is executed to retrieve data from a specific *database* where the data is stored. A *database* could be a collection of files, a relational or object database, etc. Typically, an *extractor* would receive some input (query criteria) and output the results of the query. In many cases, it may be expedient to integrate the necessary *encoders* to the *extractor* to generate delivery files that contain data in *standard formats*.

The information that the *data extractor* needs, including the user *request* input, is contained as key/value pairs in the template file `extract.tmpl`. This file is read and converted into a perl associative array, which is a named list of key-value pairs arranged so that each key is associated with its value. The MRSS creates this array and calls it `%value`. To perform the conversion from `extract.tmpl` to `%value`, the MRSS uses an algorithm that concatenates the related parameter names in `extract.tmpl` into a set of distinct parameter names for `%value`. Since each "new" *parameter* (key) retains its matching *value*, it is a simple matter to load the associative array.

There are several ways of developing an extraction program that can access the key-value pairs in `%value` to obtain the input parameters for extracting the requested data. The simplest method is to write the key-value pairs to a *control file* and direct the extraction program to read this file for input. Another method is to pass the necessary key-value pairs to the extraction program as command-line arguments. Still another method is to link the extraction routines (written in C or Perl) to the MRSS extractor module and pass the parameters internally.

> **Caution**: This method requires extensive knowledge of Perl programming languages.

The *data extractor* must output the data to files in the `$DATA_DIR/<Request_ID>` directory, and that directory's path must be passed to the *data extractor* via the command line or *control file*.

Linking local data extractors to the MRSS is a two-step process:

a. Develop a data extractor program that extracts data from the user's *request* and outputs it to a file in the `$RSS_DIR/<Request_ID>.` Choose one of the two following methods:

   (1) Pass the input values as *command-line arguments*: Create a Perl function to access the `%value` array and invoke the *data extractor* with command-line arguments, and then integrate the function into the MRSS by coding it as a function call from the library file `Data-resolution.pl`, which is located in the `$RSS_DIR/lib` directory. The function should call the extraction program as a system call. Example routines are available from MEL support.

   (2) Pass the input values via a *control file:* Use the database parameters from the associative array `%value` to construct a short Perl function creating a *control file*. (Suggestion: store the *control file* in the `$DATA_DIR/<Request_ID>` directory where the data files are stored.)

b. Create a short script or function to create a text file called `FILES` (`$DATA_DIR/<Request_ID>/FILES`), which contains the list of data files (generated by the *data extractor*) that are to be delivered to the user. The *delivery* module reads this file to prepare the list of files for delivery.

# SECTION 8. MRSS STANDARDS AND DATA FORMATS

## 8.1 FGDC CONTENT STANDARD METADATA

The following paragraphs were taken from the FGDC Content Standards for Digital Geospatial Metadata:

> *On April 11, 1994, President Clinton signed Executive Order 12906, Coordinating Geographic Data Acquisition and Access: The National Spatial Data Infrastructure. This executive order instructs Federal agencies to use the standard to document new geospatial data beginning in 1995, and to provide these metadata to the public through the National Geospatial Data Clearinghouse.*
>
> *The standard specifies information that helps prospective users to determine what data exist, the fitness of these data for their applications, and the conditions for accessing these data. Metadata also aid the transfer of data to other user's systems.*

### 8.1.1 DEFINING A METADATA RECORD

A metadata record can describe one dataset (one two-dimensional grid) or a collection of datasets. In determining what granularity is best for your site, consider that by describing the largest collection of datasets that share a unique set of properties you can reduce the number of metadata records your site needs to create and maintain. These properties include geospatial coverage, date/time range, and type of data. For example, one metadata record can describe a 30-day collection of daily numerical weather prediction model output for a particular area.

### 8.1.2 MEL METADATA FORMAT

The metadata standard is a content standard — it specifies what a metadata record contains, not what it looks like. All MEL metadata records must be in a physical format that will pass through the MEL metadata compiler error-free. This is critical, because the MEL Access Sites use the compiler to create HTML output for users to view. The MEL syntax is used to describe users' choices in ordering data: date, time, region, parameter, level, forecast period, satellite channel, etc. The MEL format is described at:

`http://geochange.er.usgs.govpub/tools/metadata/tools/doc/encoding.html`

The MEL Web site includes the Metadata Validation Service, which provides interactive help to MEL Regional Sites to validate their metadata records using the MEL metadata compiler. It is available at:

*http://www-mel.nrlmry.navy.mil/mel-bin/meta-val*

All new Regional Sites that have not already installed a *free-WAIS* server are encouraged to make use of this facility to host their metadata record at the MEL/AS until Isite is readily available.

The HDF library at the MEL Web site includes both C and FORTRAN interfaces; the BUFR and GRIB libraries were developed at Naval Research Laboratory in Monterey as C interfaces. Links to all encoder/decoder libraries are available at:

*http://www-mel.nrlmry.navy.mil//software.html#encoders_decoders*

MEL Regional Sites are encouraged to use these libraries for encoding their datasets. The Center for Air Sea Technology (CAST) has developed a GRIB Viewer[4] which converts GRIB files to NETCDF files. It uses the NcView software to display rasterized images of the gridded fields.

The standard formats currently in use are:

**GRIB**

GRIB is the World Meteorological Organization (WMO) format for Gridded Binary data. MEL uses GRIB for encoding gridded data fields for atmospheric and oceanographic datasets.

**BUFR**

The WMO Code Form FM 94, Binary Universal Form for the Representation of meteorological data (BUFR), is a binary code designed to represent and encode data elements in a continuous binary stream. MEL uses BUFR to encode observations and data such as station and ship reports.

**HDF[5]**

The Hierarchical Data Format (HDF) is a self-defining file format for transfer of various types of data between different platforms. HDF was developed by the NCSA, at the University of Illinois in Urbana-Champaign to enable the

[4] The current version does not work well with MEL GRIB Library software versions 2.0 and higher. The viewer will become fully compatible when GRIB Library 3.0 is released.
[5] Use of HDF in MEL requires prior approval and is not meant to be a substitute for GRIB where applicable.

transportation and storage of scientific data, in particular, the output from supercomputer simulation results. Currently, HDF is used to deliver only derived satellite sensor data where 8-bit encoding is not possible.

Information about a particular type of data is grouped into datasets such as raster image sets, scientific datasets, or vector sets. Each set defines an application area supported by HDF. By defining sets, new sets can be added to HDF when the need arises.

The HDF library contains both C and FORTRAN interfaces for storing and retrieving compressed or uncompressed raster images with palettes, and an interface for storing and retrieving n-dimensional scientific datasets together with information about the data, such as labels, units, formats, and scales for all dimensions. HDF is the official data format for imagery data for the Earth Observing System – Data Information System (EOSDIS) project. It is a de facto standard, and many visualization applications read data in this format. Current versions of HDF support the Network Common Data Format (NETCDF) format as well. MEL uses HDF for imagery data. HDF source code and documentation are available at:

*ftp://ftp.ncsa.uiuc.edu/HDF*

Some general information on HDF, including a Frequently Asked Questions (FAQ) page, is available from:

*http://hdf.ncsa.uiuc.edu*
*http://www.ncsa.uica.edu/SDG/Software/HDF/HDFIntro.html*

**NITF**

The National Imagery Transmission Format (NITF) is a DoD standard for imagery products. It is a transfer format for imagery products and can include images as well as subimages, symbols, labels, text, and other information that relate to the image. It is a 'wrapper' around other common formats, like Joint Photographic Experts Group (JPEG) and Graphic Information File (GIF). A display application is available, but it does not run on many platforms and can be difficult to compile. It will only display some of the formats that can be included in an NITF file. Raw satellite data using floating point values need to be stored as an extension field, which cannot be viewed with the viewer application. The NITF home page is available at:

*http://164.214.2.59/imagery/NITFS/*

# SECTION 9. TROUBLESHOOTING

## 9.1 INSTALLATION ERROR MESSAGES

### 9.1.1 IF YOU RECEIVED A *REQUEST-SITE* ERROR MESSAGE, FOLLOW THESE SUGGESTIONS:

Run `perl` -c on `$RSS_DIR/bin/mel_rrh` and `$RSS_DIR/lib/header.pl` to detect syntax errors which may have been entered during configuration. Use the following commands:

```
$ cd $RSS_DIR/bin

$ perl -c  ./mel_rrh

$ cd $RSS_DIR/lib

$ perl -c header.pl
```

If you find an error, correct it.

If this does not solve the problem, check the permissions on the `$RSS_DIR` and `$RSS_DIR/tmpl` directories and the *JCB* file. User `'mel'` should have write permission on both directories and the *JCB* file.

To change permissions (example):

```
$ chmod u+w <JCB_file>
```

### 9.1.2 IF YOU HAVE NO NEW DIRECTORY AND/OR FILES:

Check all paths you configured in the `header.pl` file.

Check permissions on `$RSS_DIR/tmpl`/. Be sure user 'mel' has write permission.

To change permissions:

```
$ chmod u+w ./tmpl
```

Check your e-mail. The MEL Regional Site Software may have sent a message identifying the errors. Look in the **error list** for a description of the identified error.

## 9.2 MRSS TROUBLESHOOTING

### 9.2.1 REQUEST STATE TRANSITIONS

While the MRSS itself has no inherent *states*, each *request*, as it moves through the system, passes through several well-defined *states*. Understanding them helps in troubleshooting an MRSS installation.

The previous section on **MRSS Architecture** contains helpful information on understanding the creation and management of state information.

Generally, if a job is not processed, check the following:

- Did the user get any confirmation? Obtain a copy of the e-mail text, if the user had saved it.
- What was the Request_ID?
- Did the request reach MRSS? Check the system logs to see if there are any errors related to this.
- If you receive a PGP encrypted mail from the mel_rrh, try decrypting it using the *PGP Manager*. It may provide valuable clues to solving the problem.
- Is the *scheduler* running? The script `$RSS_DIR/contrib/check_status` can be used to automatically check server status and restart if necessary.
- What is the effective UID of the *scheduler*? It should have read/write permissions to the `JCB` and `$RSS_DIR/tmpl` directory and any files and directories associated with the extractors.
- Is/Was the job in some `ERROR_*` state? Refer to the various error states and possible causes for them.
- If the delivery module could not deliver data, check for a valid FTP URL and also verify the existence of the destination directory and proper write permission.
- If the extractor did not find any data, check the user input conditions for validity. Operational data could have been purged.
- Are all the necessary environment variables (such as LD_LIBRARY_PATH, database environment variables, etc.) being explicitly pushed in at run-time?

> **NOTE**: Do NOT depend on the login files to set up the runtime environment in non-interactive/batch mode.

If you still are not able to solve the problem, collect as much information about the problem as possible and e-mail a description of the problem to:

*mel_support@nrlmry.navy.mil*

# SECTION 10. SITE ADMINISTRATION

## 10.1 USING THE ADMINISTRATION TOOL

The *MRSS Admin Tool* is the Site Administrator's primary utility package. It requires the use of a Web server and a Web browser. Its uses include:

- Edit the `header.pl` file to change the MRSS configuration
- Monitor activity at the site via the contents of the JCB
- Hold or terminate a request
- Manage the DAM ACL *(future function)*

Because it is a very powerful tool, the *MRSS Admin Tool* has restricted access. Two access requirements exist:

- The IP Address of the computer using the *MRSS Admin Tool* must be approved for access.
- A valid password must be used to invoke the tool.

> **NOTE:** Approved IP address/password combinations are located in `$RSS_DIR/lib/mel_admin_users.list`.

## 10.2 MRSS ACCESS CONTROL

### 10.2.1 ACCESS CONTROL MODULE

The *Access Control Module (ACM)* is a set of sub-modules and utility functions that can be used by other MRSS modules to *authenticate* and *authorize* user access to a Regional Site's resources (data and information). Since the ACM is modular, Regional Sites may replace any of the modules and/or functions to suit their requirements. For instance, a Regional Site could use its own access control module that it prefers to use instead of the MRSS DAM.

### 10.2.2 CLIENT AUTHENTICATION MECHANISM

The CAM uses the PGP software to ensure only e-mail requests from pre-approved clients are processed. It uses the MEL PGP Library to perform digital signature checks against the incoming PGP encrypted e-mail and the PGP keys in the *keyring*.

### 10.2.3 DATA AUTHORIZATION MECHANISM

The *Data Authorization Mechanism (DAM)* provides a flexible, yet powerful, means of limiting access to the local databases. Each MRSS user is identified by a unique e-mail address, which is a key attribute in the DAM *Access Control List (ACL)*. This list is maintained in a text file that should be accessible only by the MRSS software, the MRSS Site Administrator, and local users such as the Site Database Administrator(s).

Key features of the DAM include:

- grouped and individual authorization
- optional password protection[6] for any or all datasets
- password expiration
- optional FTP domain and PGP key requirements
- access limited to individual users and/or Internet domains

The DAM allows local databases and datasets to be grouped logically. A *database* may consist of one or more *datasets*. Local sites can define their own *database* and the *datasets*, but the definition must be common to both the DAM and *MRSS Extractor*, which may also use this classification.

Access to a *dataset* that is part of a given *database* is evaluated according to a set of *rules* and *flags* and pre-defined *user profiles* that each apply to the *database* and *dataset* domain. Both *rules* and *flags* are sets of conditions that can be applied to input information about the user (such as identification and affiliation). A *flag* can be used to set up a optional requirement or condition and can assume one or more values, but only from a pre-defined set of values, such as YES or NO. For simplicity, a given *flag* may be set only once in a domain. *Rules* may be repeated as often as necessary to define requirements for access or denial. The following list identifies and defines valid *rules* and *flags*:

**Rules:**

```
Order [Allow,Deny | Deny,Allow]
   allow [from] <full_or_partial_domain | user_id>
   deny [from] <full_or_partial_domain | user_id>
```

[6] The MEL Access Site Software has not yet implemented this support for MEL users. A user will not be able to enter a password via the order form. However, the Regional Sites may support password usage in orders that are received directly from the user, for example, to support product generators.

**Flags:**

```
D_Default [Allow | Deny]

PasswordRequired  [Yes | No]

DefaultFTPrequired [Yes | No]

PGPkeyRequired [Yes | No]
```

**User Profile:**

```
<user_id:encrypted_password:default_FTP_root:expiry_date>
```

**Definitions of DAM Rules and Flags:**

`D_Default`

This rule handles the cases for default action (to allow or to deny access) when a request could not be resolved, for example, when the *database* or *dataset* exists in ACL. Note, that before setting this flag to `Allow`, free access can be provided to those datasets that do not require access control without the need to maintain them in ACL. On the other hand, if this flag is `Deny`, then all datasets need explicit rules for access.

`PasswordRequired`

All access to this *database+dataset* requires a valid password. The *request* would be denied immediately if the user has no password.

`DefaultFtpRequired`

Data will be sent ONLY to a pre-authorized `FTP` server. The user has to specify an FTP server address while requesting authorization in advance. This default FTP server will be specified in the user profile.

`PGPkeyRequired`

Data will have to be encrypted; user has to provide a certified and locally verified PGP public key while making the request. The key must be available in the MRSS PGP keyring.

`Order`

Specifies the precedence for evaluating rules. It is followed by either:

```
Allow,Deny
```

`Deny,Allow`

This defines which set of rules are to be evaluated first.

`Allow [from]`

Access allowed unless denied later.

`Deny [from]`

Access denied unless allowed later.

`All` [or some regular expression or a fully qualified name] self explanatory.

**General Set of Default Conditions**

For access, a user's *request* must satisfy the *rules* and *flag* conditions for both these domains.

If no rules are defined for a ***`database+dataset`*** pair, access is denied by default. This default can be changed to '`allow by: ?????`.' This acts as a place holder and decreases the chance of data being put online by mistake. Even if no *rules* or *flags* are defined for this pair, there must be an entry for it in the *authorization list*.

For every ***`database`***, *rules* may be defined that could affect all ***`datasets`*** within this group.

If a ***`database`*** domain *rule* or *flag* causes the *request* to be denied, the ***`dataset`*** domain *rules* and *flags* are ignored. The ***`database`*** domain *rules* and *flag* conditions must be satisfied first, or else the *request* access will be denied. This default can be changed.

The *Admin Tool* provides an interface to manage this list (in progress).

A sample *access control* file follows. Note that indentation and capitalization is strictly for the user's benefit. The file can be in any format, and all *rule* and *flag* keywords are case-insensitive, as are database and dataset names. Comments are preceded by the symbol #.

SAMPLE *ACL* FILE

```
#
# Sample MRSS DAM access-control file
# Several rules are defined for FIRST_DSET, while
# access to SECOND_DSET is unlimited.
#
```

```
<DATABASE = Database_1>
  #
  # database domain rules follow
  #

  #
  # with these rules, users from any .com site
  # except those from "badguys.com" will be allowed to
  # order data from this database
  #

  Order allow, deny
  deny from badguys.com
  allow from *.com

  <DATASET=FIRST_DSET>

    #
    # users desiring to get this dataset will
    # have to already have a PGP key registered
    # with this regional site
    #

    PGPKEYREQUIRED = yes

    #
    # users desiring to get this dataset will
    # have to have an entry for them added to this
    # site's access control list with a valid password
    #

    PASSWORDREQUIRED=YES

    #
    # dataset rules
    #

    Order deny, allow

    #
    #  deny access from all sites, except
    #  cast.msstate.edu, and all *.mil sites
    #

    deny from all
    allow from cast.msstate.edu
    allow from *.mil

    #
    # user level rules
    # this is where passwords and default FTP address
    # are stored
    #

    Allow from
<rms@cast.msstate.edu:*zxKtJX4dCwYK2:ftp.cast.msstate.edu/pub/incoming:19981229>
    Allow from
<billy@cast.msstate.edu:zxKtJX4dCwYK2:ftp.cast.msstate.edu:19981230>
    Allow from
<stein@nrlmry.navy.mil:kfddIId83J:helium.nrlmry.navy.mil/pub/receive/stein:19971030>
```

```
  </DATASET>  #  Dataset_1

  <DATASET=Dataset_2>
    allow all
  </DATASET>  #  Dataset_2

</DATABASE>  #  Database_1

#
# end of sample access-control list
#
```

## 10.3 MANAGING SUBSCRIPTIONS

### 10.3.1 SUBSCRIPTIONS

The MRSS subscription service allows data to be automatically distributed to the user every time it becomes available. Any data that is regularly updated (such as, weather forecasts, satellite data, etc.) are candidates for the subscription service.

To support this service at the Regional Site, a subscription *trigger* is executed when new data becomes available. This *trigger* processes specific data requested in the active subscriptions. The recommended procedure is for triggers to be initiated by an ingest process for the specific data (if it exists). This guarantees the subscription will be serviced only when new data is available. Alternatively, a trigger can be called by a UNIX `crontab` job scheduled at a regular interval, but it is important that the trigger be called only when new data is available. If a trigger is called when new data is not ingested, the potential exists for the same data to be delivered again.

Included in the MRSS is a generic trigger program called `mel_sub_trigger`. Usage is:

```
mel_sub_trigger db_name dset_name sub_dir [tag_1 value_1 [tag_2
value_2 [...]]]
```

| **Argument** | **Definition** |
|---|---|
| `db_name` | Database name associated with the dataset the name of the dataset |
| `dset_name` | Name of the dataset |
| `sub_dir` | Directory path from $RSS_DIR/subscriptions/db_name/dset_name |
| `tag_n` | Unique string used to identify a specific line in the subscription request file to replace |
| `value_n` | Replacement Value or string applied to line specified by `tag_n` |

The arguments db_name and dset_name are used by various modules and the subscription mechanism. These two arguments are also used in the LOCAL section of the metadata choice specification to identify the dataset. The argument sub_dir is also used in the LOCAL section of the metadata choice section to identify where the subscription request should be kept (that is, $RSS_DIR/subscriptions/db_name/dset_name/sub_dir). db_name and dset_name may also optionally be used in the **Data Access Module** for restricting access to data.

To enable subscriptions for a dataset, the REQUEST_TYPE field in the metadata choice section of the metadata must include the word SUBSCRIBE. The typical line for subscription-enabled data would read:

```
REQUEST_TYPE = GET, SUBSCRIBE
```

A user subscribes to a dataset by choosing “subscribe” from a drop-down list on the order form. The order processing service at the MEL Access Site sets the REQUEST_TYPE field in the order string equal to SUBSCRIBE. When the SUBSCRIBE order is received and processed by the *mel_rrh module*, the request is placed in the file and the order text is stored in the file $MEL_RRS/subscriptions/$db_name/$dset_name/$sub_dir/$request_id. If the directory does not exist, it is created automatically.

When the trigger is executed, the subscription directory $RSS_DIR/subscriptions/$db_name/$dset_name/$sub_dir is searched for subscription orders. If any exist, they are processed, one at a time, line by line. It starts by replacing the first line with a regular expression match for each tag with associated values included on the trigger command line, replacing REQUEST_ID with SUBSCRIPTION_ID, setting CONFIRM = NONE, and then submitting it as a new order to the *mel_rrh module*. When the *mel_rrh module* receives an order with a SUBSCRIPTION_ID it creates a new request_id of the form: $SITE-xnnnnnn , where "x" indicates subscription and nnnnnn is a unique number. It is then processed like any other order in the MEL system. Any error messages generated by the mel_sub_trigger program are e-mailed to the MEL_ADMIN mail list.

### 10.3.2 CUSTOM SUBSCRIPTION TRIGGERS

There are times when a custom trigger might be required for a particular dataset. In this case you can make a copy of the mel_sub_trigger program, rename it, and customize it for your needs. One example of this is the need to conditionally process subscriptions depending on the subset choices made in the order. For instance, only process the subscription if the forecast periods specified in the order are less than a given value passed in to the new trigger program. This is a real example based on data

that is ingested over a period of time (earlier forecasts first) and the desire is to deliver data as soon as possible. In this case the trigger program was modified to accept the additional parameter of a forecast period. The trigger program only submits orders for subscriptions that contain orders for forecasts less than the given value. The ingest routine calls the *trigger* after each set of forecast fields is received. This method provides timely delivery of data.

## 10.4 MRSS GENERIC FILE EXTRACTOR INTERFACE

The MRSS *Generic File Extractor (GFE)* is a special extractor class where all the necessary information is contained in the pathnames and filenames of the datasets. The extractor uses a special field in the <LOCAL> tag of the e-mail message and any necessary pieces of information from the sub-setting sections to construct the unique paths necessary to refer to the relevant files. These files are then copied over or linked to the MRSS delivery directory.

> **NOTE:** More specific information on the Generic File Extractor will be added in the next version of this manual.

## 10.5 MRSS SUPPORT

### 10.5.1 SITE MAINTENANCE TASKS

**Daily maintenance**

- backup schedule and procedure
- system function checklist
  - error log
  - scheduler operation
  - mail delivery
  - security procedure

**Long-term maintenance**

- server alias
- directory structure
- use of relative URL's
- subscribe to the `mel_support@nrlmry.navy.mil` e-mail list.

# APPENDIX A. LIFE CYCLE OF A *REQUEST*

## THE *REQUEST* IS GIVEN AN IDENTIFIER AND SENT FROM A MEL ACCESS SITE TO A REGIONAL SITE

When a user requests data from a Regional Site via a MEL Access Site, the *request* is first assigned a unique *Request Identifier* (`Request_ID`). The Request_ID is constructed using an access site prefix (which is null right now), regional site prefix, and a numeric field based on time. Then the *request* is sent to the Regional Site via e-mail. The `Request_ID` is used to track the progress of the *request*, create a unique directory structure for storing temporary files and data before delivery, and log the status of the process in the *JCB*. In the following description, the `Request_ID "site-123456"` will be used in the examples.

## THE REGIONAL SITE REQUEST-HANDLER PICKS UP THE REQUEST

The Regional Site *request-handler* (*RRH*) picks up the request and looks for the `Request_ID` in the *subject* field of the e-mail message. If the *request* originated from a source other than a MEL Access Site, the *RRH* assigns a `Request_ID` .

The *RRH* decrypts it using the Regional Site's password and private key. It then authenticates the origin of the message by checking its signature against signatures on the local PGP *keyring*. The public keys of the MEL Access Sites must therefore be on the *keyring*.

After *decryption* and *signature authentication*, the *RRH* parses the message and activates the parser which converts the message into a perl multidimensional associative array, the values of which can be referenced using the proper keys. Next, it creates a set of *template files*, each containing only the information needed by the modules that intend to access their respective template files. It is through these *template files* that the relevant portions of the e-mail message are conveyed to other functional modules such as the *extractor* and *delivery* modules. If the system were to crash, the information contained in the template files and the JCB would allow the system to restart at the beginning of its last state.

Next, the *RRH* sends an e-mail acknowledgment, if requested, to the user confirming the message was received and identifying the *Request Identifier* for future reference.

Finally, the *RRH* adds an entry to the *JCB* for this *request*, assigning it a status of `AWAITING-ACCESS` if *automatic* access control is enabled.

If any of these processes fail, the *RRH* sends the *request* via e-mail to the Regional Site Administrator.

## THE *SCHEDULER* PLAYS A CENTRAL ROLE IN HANDLING THE REQUEST

After the *RRH* makes an entry in the *JCB*, the *scheduler* selects `site-123456`, and invokes the appropriate modules in the following sequence:

a. The *scheduler* notifies ACM/DAM (`mel_access`) to process `site-123456` and updates the *status* of `site-123456` in the *JCB* file to IN_ACL. The user name and dataset requested are compared to the *authorization list*. If the user is allowed access to that data, DAM updates the status of `site-123456` to `AWAITING_EXTRACTION`.

b. The *scheduler* notifies the *extractor* to process `site-123456`, and updates the status of `site-123456` in the *JCB* file to `IN_EXTRACTION`. The *extractor* then reads the *file* `$TMPL_DIR/site-123456/extract.tmpl` to obtain the parameters for extracting the requested data. Once the data has been extracted, the *extractor* places it in `$DATA_DIR/site-123456` and updates the *JCB* file to status `AWAITING_POSTPROC`.

c. The next time the *scheduler* selects `site-123456`, it notifies the *delivery module* to deliver `site-123456`. It then updates the *status* of `site-123456` to `IN_DELIVERY`. The delivery program then reads the file `$TMPL_DIR/site-123456/delivery.tmpl` to find the delivery address and delivery method. Finally, it delivers the files listed in the `$DATA_DIR/site-123456/files` to that address and updates the *status* of `site-123456` in the JCB file to `AWAITING_CLEANUP`.

When the *scheduler* selects `site-123456` for the final time, it runs the *cleanup* function to delete the entry and all associated temporary files.

# APPENDIX B. MRSS FILE LISTING

## INVENTORY OF SOFTWARE CONTENTS

In the listings that follow, filenames are shown first, followed by their size (in bytes), and their date and time of creation in 1997. Listings in the File Name column that end in "/" are subdirectories.

Some subdirectories are intentionally empty; the following directories are for MRSS runtime files.

In the MRSS top-level directory ($RSS_DIR):

| File Name | Size | Date | Time |
|---|---|---|---|
| CHANGES | 1345 | 4/21 | 15:49 |
| README | 307 | 4/21 | 15:49 |
| VERSION_1.1.00 | 75 | 4/21 | 15:49 |
| bin/ | 512 | 4/22 | 13:27 |
| cgi-bin/ | 512 | 4/21 | 15:49 |
| contrib/ | 512 | 4/21 | 15:49 |
| data/ | 512 | 4/21 | 15:49 |
| doc/ | 512 | 4/21 | 15:49 |
| inspath | 738 | 4/21 | 15:49 |
| install_rss | 2760 | 4/21 | 15:49 |
| lib/ | 512 | 4/21 | 15:49 |
| log/ | 512 | 4/21 | 15:49 |
| runtime/ | 512 | 4/21 | 15:49 |
| src/ | 512 | 4/21 | 15:49 |
| subscriptions/ | 512 | 4/21 | 15:49 |
| tmpl/ | 512 | 4/21 | 15:49 |

In Subdirectory `./bin`:

| File Name | Size | Date | Time |
|---|---|---|---|
| mail/ | 512 | 4/21 | 15:49 |
| mel_access | 3491 | 4/21 | 15:49 |
| mel_cleanup | 3984 | 4/21 | 15:49 |
| mel_deliver | 9490 | 4/21 | 15:49 |
| mel_extract | 4694 | 4/21 | 15:49 |
| mel_postproc | 4963 | 4/21 | 15:49 |
| mel_rrh | 14347 | 4/21 | 15:49 |
| mel_scheduler | 15284 | 4/21 | 15:49 |
| rss_patch | 1785 | 4/21 | 15:49 |
| showjcb | 2182 | 4/21 | 15:49 |
| subread | 1335 | 4/21 | 15:49 |

In Subdirectory `./bin/mail`:

| File Name | Size | Date | Time |
|---|---|---|---|
| test.mail | 3103 | 4/21 | 15:49 |

In Subdirectory `./cgi-bin`:

| File Name | Size | Date | Time |
|---|---|---|---|
| AdminTool | 423 | 4/21 | 15:49 |
| DAM_main.cgi | 4467 | 4/21 | 15:49 |
| access.cgi | 3611 | 4/21 | 15:49 |
| adduser.cgi | 1109 | 4/21 | 15:49 |
| default.cgi | 548 | 4/21 | 15:49 |
| delete.cgi | 1743 | 4/21 | 15:49 |
| delete_ver.cgi | 3149 | 4/21 | 15:49 |
| index.cgi | 3542 | 4/21 | 15:49 |
| jcb_1a.cgi | 5252 | 4/21 | 15:49 |
| jcb_2.cgi | 4065 | 4/21 | 15:49 |
| meljobs.cgi | 1115 | 4/21 | 15:49 |
| pgpmgr | 40087 | 4/21 | 15:49 |
| readnew | 7215 | 4/21 | 15:49 |
| reconfigure2.cgi | 5132 | 4/21 | 15:49 |
| reconfigure2_update.cgi | 763 | 4/21 | 15:49 |
| savuser.cgi | 1259 | 4/21 | 15:49 |
| screens | 2750 | 4/21 | 15:49 |
| stats.cgi | 469 | 4/21 | 15:49 |
| translog.cgi | 2054 | 4/21 | 15:49 |
| update.cgi | 389 | 4/21 | 15:49 |
| update2.cgi | 2078 | 4/21 | 15:49 |

In Subdirectory `./contrib`:

| File Name | Size | Date | Time |
|---|---|---|---|
| EXAMPLES | 415 | 4/21 | 15:49 |
| README | 362 | 4/21 | 15:49 |
| check_status | 4479 | 4/21 | 15:49 |
| check_status.eg | 719 | 4/21 | 15:49 |
| inc.pl | 80 | 4/21 | 15:49 |
| movelogs | 549 | 4/21 | 15:49 |

In Subdirectory `./data`:

| File Name | Size | Date | Time |
|---|---|---|---|
| usmail/ | 512 | 4/21 | 15:49 |

In Subdirectory ./data/usmail:

In Subdirectory ./doc:

| File Name | Size | Date | Time |
|---|---|---|---|
| html/ | 512 | 4/22 | 13:26 |

In Subdirectory ./doc/html:

| File Name | Size | Date | Time |
|---|---|---|---|
| graphics/ | 512 | 4/21 | 15:49 |
| meldocs.html | 7747 | 4/21 | 15:49 |

In Subdirectory ./doc/html/graphics:

| File Name | Size | Date | Time |
|---|---|---|---|
| A097.jpg | 2934 | 4/21 | 15:49 |
| AT_WORK_.GIF | 271 | 4/21 | 15:49 |
| BUTTON1-TRANS.GIF | 2300 | 4/21 | 15:49 |
| CRYSTAL_B.GIF | 599 | 4/21 | 15:49 |
| GLOBE.GIF | 6471 | 4/21 | 15:49 |
| LTBKGRND.GIF | 4244 | 4/21 | 15:49 |
| binoc_bt.gif | 2206 | 4/21 | 15:49 |
| codebtn.gif | 260 | 4/21 | 15:49 |
| greybox.gif | 1927 | 4/21 | 15:49 |
| key.gif | 268 | 4/21 | 15:49 |
| large_mel_logo.gif | 23557 | 4/21 | 15:49 |
| log.gif | 1187 | 4/21 | 15:49 |
| magnifying_g.gif | 343 | 4/21 | 15:49 |
| manual.gif | 437 | 4/21 | 15:49 |
| reconfigure.gif | 260 | 4/21 | 15:49 |
| restart.gif | 305 | 4/21 | 15:49 |
| stats.gif | 538 | 4/21 | 15:49 |
| stats_gr.gif | 299 | 4/21 | 15:49 |
| tool.gif | 217 | 4/21 | 15:49 |

In Subdirectory `./lib:`

| File Name | Size | Date | Time |
|---|---|---|---|
| Asso.pm | 701 | 4/21 | 15:49 |
| DAM.pm | 14367 | 4/21 | 15:49 |
| Data_resolution.pl | 1723 | 4/21 | 15:49 |
| Logger.pm | 6054 | 4/21 | 15:49 |
| acl.pl | 3633 | 4/21 | 15:49 |
| array_DAM.pm | 56486 | 4/21 | 15:49 |
| err_mail.pl | 1685 | 4/21 | 15:49 |
| extract_grid.pl | 3185 | 4/21 | 15:49 |
| ftp.pl | 25942 | 4/21 | 15:49 |
| ftp_util.pl | 4046 | 4/21 | 15:49 |
| gpe.pl | 6725 | 4/21 | 15:49 |
| header.old | 973 | 4/21 | 15:49 |
| header.pl | 1299 | 4/21 | 15:49 |
| help.pl | 3027 | 4/21 | 15:49 |
| jcb.pl | 7567 | 4/21 | 15:49 |
| lchat.pl | 6199 | 4/21 | 15:49 |
| logging.pl | 617 | 4/21 | 15:49 |
| mel_admin_users.list | 437 | 4/21 | 15:49 |
| mel_util.pl | 3673 | 4/21 | 15:49 |
| parser.pl | 1003 | 4/21 | 15:49 |
| parsing.pl | 8145 | 4/21 | 15:49 |
| pgp.pl | 9653 | 4/21 | 15:49 |
| pgp2.pl | 22645 | 4/21 | 15:49 |
| pgppass.pl | 252 | 4/21 | 15:49 |
| redirect.pl | 140 | 4/21 | 15:49 |
| rss_example.pl | 1304 | 4/21 | 15:49 |
| signal_handler.pl | 439 | 4/21 | 15:49 |
| subscribe.pl | 3303 | 4/21 | 15:49 |
| templates.pl | 5319 | 4/21 | 15:49 |

In Subdirectory `./log`:

| File Name | Size | Date | Time |
|---|---|---|---|
| history/ | 512 | 4/21 | 15:49 |
| msg/ | 512 | 4/21 | 15:49 |
| trans/ | 512 | 4/21 | 15:49 |

In Subdirectory `./log/history`: Empty

In Subdirectory `./log/msg`: Empty

In Subdirectory `./log/trans`: Empty

In Subdirectory `./runtime:`

| File Name | Size | Date | Time |
|---|---|---|---|
| DAM.list | 503 | 4/21 | 15:49 |
| JCB | 0 | 4/21 | 15:49 |
| jobnumber | 0 | 4/21 | 15:49 |
| schedule.dat | 1529 | 4/21 | 15:49 |

In Subdirectory `./src`:

| File Name | Size | Date | Time |
|---|---|---|---|
| README | 884 | 4/21 | 15:49 |
| schedule.dat | 1450 | 4/21 | 15:49 |
| subtime.ansi | 6670 | 4/21 | 15:49 |
| subtime.c | 7074 | 4/21 | 15:49 |
| subtime.c.bak | 7074 | 4/21 | 15:49 |
| subtime.kr | 6670 | 4/21 | 15:49 |

In Subdirectory `./subscriptions`:

| File Name | Size | Date | Time |
|---|---|---|---|
| RSS_EXAMPLE_DSET/ | 512 | 4/21 | 15:49 |

In Subdirectory `./subscriptions/RSS_EXAMPLE_DSET`: Empty

In Subdirectory `./tmpl:` Empty

# APPENDIX C. MEL E-MAIL ORDER SYNTAX

*Version 1.1 (Final Release 2 June 1997 - VGA/CAST)*

The *MEL Regional Sites (MEL/RS)* receive the request for the locally provided datasets via e-mail sent typically from one of the official *MEL Access Sites (MEL/AS)*. Alternately, it is also possible for an authorized user to send an e-mail directly to the Regional Site. However, the messages must conform to the MEL E-mail Order Syntax (hereafter referred to as **syntax**) defined in this appendix since these e-mail messages are processed automatically by the local *MEL Regional Site Software (MRSS)*. Also, the messages should be PGP encrypted using the regional site's public key.

The order message has *SGML-like* syntax. Although the syntax is not fully SGML compliant, plans call for a fully SGML compliant syntax with a corresponding *Document Type Definition (DTD)*. For an introduction to SGML see:

*http://etext.virginia.edu/bin/tei-tocs$div=DIV1&id=SG*

A **MEL E-mail Order Message** (hereafter simply referred to as **message**) is made up of TAGS and KEYWORDS**.** In its scope, a TAG may include zero or more KEYWORDS and/or other TAGS. A TAG can be considered a section header in the message. KEYWORDS have a VALUE associated with them. The special KEYWORD "NAME" is associated with a TAG to associate a single VALUE with a TAG. Therefore this particular KEYWORD should not be used except along with a TAG. All TAGS must have a beginning and end. TAGS are enclosed in angle brackets, such as <TAG>. An end TAG is of the form </TAG>. TAGS can be nested within one another. All KEYWORDS are required to have one or more associated VALUES.

All TAGS, KEYWORDS, and VALUES are case-sensitive (this could change in the next release). The message can contain comments which are considered free text, and are ignored by the parser. Comments can begin anywhere in a line starting from the first occurrence of the character "#". All text following the "#" character is ignored. Multiline comments are not allowed, and comments do not continue into the next line, unless the next line begins with "#" which indicates a new line comment. The following characters are reserved: "#" "<", ">", "/". There are no escape sequences available for these reserved characters. Long lines can be broken down using the line continuation character "\". In the following examples, optional fields are contained in square brackets […].

A TAG has the form:

```
<TAG [NAME = Tag_VALUE]>
#-- and this is a comment
#-- a line of comments starts with the pound '#' sign
</TAG>
```

where NAME is the special KEYWORD that can be used to associate a VALUE with the TAG (and the NAME KEYWORD is optional).

Similarly, a KEYWORD (with-in a TAG) has the form:

```
<TAG [NAME = Tag_VALUE]>

KEYWORD = Keyword_VALUE

</TAG>
```

Here is a list of standard **TAGS** and **KEYWORDS**:

- **MEL** - This is used to indicate the beginning and the end of a MEL Order Message, with the following TAGS and **KEYWORDS** in their respective scope.
- **PROLOGUE** - This section is used to group all the general information about the user to authenticate the user and authorize the request.
  - **USER_NAME** - The name of the user. This could also be used for access control at specific sites.
  - **EMAIL** - E-mail address of the user. Messages on acknowledgment, progress, etc. will be mailed to this address.
  - **ORGANIZATION** - *(optional)* User affiliated institution.
  - **PROJECT_NAME** - *(optional)* User associated project.
  - **ST_ADDRESS1** - *(optional - mailing/shipping address may be required if DELIV_METHOD = USMAIL)*
  - **ST_ADDRESS2** - *(optional)* - Additional line to specify the street address.
  - **CITY** - *(optional)*
  - **STATE** - *(optional)*
  - **ZIP** - *(optional)*
  - **TELEPHONE** - *(optional)*
  - **FAX** - *(optional)*
  - **PGP_KEY** - *(optional)* User's PGP public key.
  - **CONFIRM** - Means of confirming the request. Default is EMAIL. In fact, e-mail is the only means of confirmation for now. The confirmation message will be sent to the user's e-mail address from above. Though support for a separate e-mail address for confirmation message sound reasonable, it is not supported at present in order to comply with the "restricted access" paradigm. Valid options are EMAIL | NONE.
  - **LIST_TYPE** - This keyword could be specified anywhere applicable and take a number of options depending upon context. This can be any applicable pre-defined or specified value(s) that are appropriate for each domain.

- **VALUES** - This is a special keyword that is associated with and always follows the KEYWORD "LIST_TYPE". The set of values will depend upon the preceding LIST_TYPE. Valid entries may include "*" denoting that this could default to whatever is applicable, determined in the local context.
- **REQUEST** - Action initiated by the request. Options include BROWSE, GET, SUBSCRIBE, UNSUBSCRIBE, HELP, etc. Each dataset online requires either BROWSE or GET. If REQUEST == SUBSCRIBE a **REQUEST_ID** will be assigned to the request. This ID will be necessary to later terminate a subscription.
- **REQUEST_ID** - *(required)*
- **DELIV_METHOD** - Means of delivering the data. Valid means include FTP | FTP_PKUP | HTTP | EMAIL | USMAIL. If USMAIL, proper mailing/shipping address may be required (ST_ADDRESS, CITY, etc.).
- **DELIV_URL** - *(required for REQUEST = GET | SUBSCRIBE or DELIV_METHOD = FTP | EMAIL - if different from default e-mail and authorized)* This is the URL in the form of `protocol://host.domain[:port]/location` where protocol can be FTP (default) or EMAIL.
- **COMPRESS** - This can be used to specify the compression algorithm to be used to reduce the size of the data files. The valid options are COMPRESS | GZIP | ZIP. The ZIP option is not supported at present.
- **TAR** - This is a packaging option with values YES | NO.
- **PGP_ENCRYPT** - Option to request PGP encryption of data files. A PGP public key for the user is required.

- **LOCAL** - A special TAG for including site specific processing information. The contents are recommended to be in NAME = VALUE pairs, similar to the use of KEYWORDS.
- **DATASET** - If the request is to perform some operation on some dataset available from the regional site, each dataset from the site is to be assigned a unique name. The dataset is used to define systematic extraction and sub-setting of its contents. A site may maintain multiple **Databases** which in turn consist of multiple datasets. The **Database** for a particular request may be specified in a LOCAL tag, if necessary.
  - **DELIV_FMT** - *(optional if supported)* By default, the data will be delivered in one of the appropriate standard delivery formats. Alternate delivery formats may be supported by the data centers.
- **ACTION** - *(one or more may be required)* This specifies some action to be performed upon the specified **DATASET**. It may include sub-setting, visualization, further analysis, etc. Methods may be nested within one another with the first specification being at the top level.
  - **DIMENSION** - *(required if ACTION = SUBSET)* This is applicable for multidimensional datasets that could be subset along the dimensions. The DIMENSIONS can be nested with the first one specified varying the slowest. If a DIMENSION is omitted, it is equivalent to specifying its valid range.

- ◊ **UNITS** - Units for the given dimension.
- ◊ **LIST_TYPE** - Specified as appropriate.
- ◊ **VALUES** - *(always need to follow LIST_TYPE)* Valid entries may include "*" denoting default determined by local context.

---

**<MEL>**

**<PROLOGUE>**

**USER_NAME = [name_of_the_user]**
**EMAIL = [e-mail_address_of_the_user]**

***ORGANIZATION = [organization, the user is affiliated with]***
***PROJECT_NAME = [project, the user is associated with]***

***ST_ADDRESS1 = [first line of street address / mail stop / etc]***
***ST_ADDRESS2 = [second line of street address / mail stop / etc.]***
***ST_ADDRESSn = [nth line ..... as many lines as needed]***
***CITY = [name of the city - for mailing address]***
***STATE = [name of the state - for mailing address]***
***ZIP = [postal zip code (US only) - for mailing address]***

***TELEPHONE = [telephone number (string)]***
***FAX = [fax number (string)]***

**CONFIRM = [EMAIL | NONE]**

**REQUEST = [GET | SUBSCRIBE | UNSUBSCRIBE]**

*REQUEST_ID = [unique identification number for each request]*

**DELIV_METHOD = [FTP | FTP_PKUP | HTTP | EMAIL | USMAIL]**
**DELIV_URL = [URL_for_delivery]**

*<LOCAL> # local keywords for the message set*
*<LOCAL_TAG>*
*local_mesg_keyword [.....]*
*..... any special extensions to local parser*
*.....*
*</LOCAL_TAG>*
*</LOCAL>*

**</PROLOGUE>**

*<LOCAL> # local keywords for the message set*
*<LOCAL_TAG>*
*local_mesg_keyword [.....]*
*..... any special extensions to local parser*
*.....*
*</LOCAL_TAG>*
*</LOCAL>*

**<DATASET NAME = [regional_dataset_name]>**

DELIV_FMT = [MEL_INTERCHANGE_FMT | LOCAL_ASCII | LOCAL_BINARY |
E2DIS | GRADS | CMES | NCARI | ...]

*<LOCAL> # local keywords for the message set*
*<LOCAL_TAG>*
*local_mesg_keyword [.....]*
*..... any special extensions to local parser*
*.....*
*</LOCAL_TAG>*
*</LOCAL>*

<ACTION NAME = SUBSET> # what are we going to do with this dataset?
<DIMENSION NAME = [name of slowest-varying dimension]>
# valid dimension names can be LATITUDE, LONGITUDE, TIME,
# FCST_TIME, DEPTH, HEIGHT, LEVEL, PARAMETER
# THETA, RADIUS, ALPHA
# X, Y, Z
*UNITS = [units]*
LIST_TYPE = SPECIFIED | RANGE | GREAT_CIRCLE
VALUES = [value1 value2 value3 .... valueN] | # if specified
[begin_val end_val increment] # for linear
# begin_ end_ increment values can be defaulted to "*",
# which means whatever is available by defaults
[start_lat, start_lon, end_lat, end_lon, delta] # GCIRC
</DIMENSION>

<DIMENSION NAME = [name of next dimension]>
*UNITS = [units for dimension 2]*
LIST_TYPE = SPECIFIED | RANGE
VALUES = [.....]
</DIMENSION>

```
    [..... for N dimensions]
          # so on till all the dimensions are specified
          # nesting of dimensions is implied
          # the final dimension varies the fastest

  </ACTION>  #  end of subsetting

</DATASET>

</MEL>  #  end of the message
```

# APPENDIX D. ACKNOWLEDGMENTS

The MEL Development Team would like to thank the following organizations and people for their support in making MEL a viable tool for worldwide users:

**DEFENSE MODELING AND SIMULATION OFFICE**

**DOD EXECUTIVE AGENTS**

- Air and Space EA
- Oceans EA
- Terrain EA

**US FEDERAL GEOGRAPHIC DATA COMMITTEE**

**FGDC NATIONAL SPATIAL DATA CLEARINGHOUSE**

This organization is responsible for coordinating the spatial data collection and dissemination activities within the United States among government and non-government organizations. Their home page is :

*http://fgdc.er.usgs.gov/fgdc.html*

**MODELING AND SIMULATION RESOURCE REPOSITORY (MSRR)**

MSRR is a collection of computer resources and information which assists the modeling and simulation community in communication and information sharing. The MSRR supports a home page at:

*http://huachuca-jdbc.army.mil/msrr/*

**REGIONAL SITES**

The software has been improved significantly using valuable feedback from the Regional Site Administrators.

Earth Observing System Data and Information System

# F

FAQ..... Frequently Asked Questions

FGDC..... Federal Geographic Data Committee

FTP..... File Transfer Protocol

# G

GB..... Gigabyte (1024 MB)

GFE..... Generic File Extractor

GIF..... Graphic Information Format

GPE..... Generic Product Extractor

GRIB..... Gridded Binary data (Encoder Ver. 2.21, Decoder Ver. 2.2)

GUI..... Graphical User Interface

GZIP..... Commercial GNU Zip file compression format product

# H

HDF..... Hierarchical Data Format, Ver. 4.0r1

HTML..... Hypertext Markup Language

HTTP..... Hypertext Transfer Protocol

# I

IP..... Internet Protocol

Isite..... Commercial integrated Internet publishing software package including a text indexer, a search engine and Z39.50 communication tools to access databases.

# J

JCB..... Job Control Block

JMCIS..... Joint Maritime Command Information System

JPEG..... Joint Photographic Experts Group image format

Relational Database Management System

ROI..... Region of Interest

RRH ..... Regional Request Handler

RSS ..... Regional Site Software

# S

SGI ..... Silicon Graphics, Inc.

SGML..... Standard Generalized Markup Language

SMTP ..... Simple Mail Transfer Protocol

# T

TAC-4 ..... Tactical Applications Computer, Series 4

TAR..... UNIX archive file format

# U

UID ..... User Identification

URL..... Universal Resource Locator

# V-W

WAIS..... Wide Area Information Service

WMO..... World Meteorological Organization

WWW..... World Wide Web

# X-Z